TRABAJO DE FIN DE CARRERA

TÍTULO: *Firewalls software*: Estudio, instalación, configuración de escenarios y comparativa

AUTOR: Mònica Ferrer Berbegal

DIRECTOR: Marco Antonio Peña Basurto

FECHA: 14 de enero de 2006

Título: *Firewalls software*: Estudio, instalación, configuración de escenarios y comparativa

Autor: Mònica Ferrer Berbegal

Director: Marco Antonio Peña Basurto

Fecha: 14 de enero de 2006

Resumen

Los usuarios de domicilios particulares y de las empresas disponen de un amplio abanico de comodidades que les permiten gestionar su trabajo a través de la red pública de Internet. Esto puede suponer un problema si no se cumplen ciertas medidas de seguridad y de control para evitar las violaciones de datos que se irrumpen en la red.

La confidencialidad de los datos adquiere mayor importancia y dificultad de mantenimiento en el ámbito empresarial, donde la comunicación se establece entre la red privada de la empresa e Internet; así pues, este proyecto se enfoca concretamente en un escenario empresarial.

Para garantizar un nivel de seguridad entre dos o más redes debe existir un sistema de aislamiento que evite comunicaciones indeseadas llamado *Firewall*. Por ello, este proyecto consiste en el estudio, la implementación y el análisis de tres tipos de *Firewall* sobre una Red Desmilitarizada o DMZ, en la cual se configuran los servicios más comunes que ofrece una empresa y que deberán ser protegidos por el *Firewall* para un acceso restringido.

Los *Firewalls* se configuran con aplicaciones de distintos sistemas operativos:

- *Iptables* bajo la distribución Debian Sarge de GNU/Linux.
- ISA Server 2004 Enterprise bajo Windows Server 2003.
- *Packet Filter* bajo OpenBSD

Una vez hecha la implementación y el análisis de las ventajas e inconvenientes de cada uno de ellos se lleva a cabo un análisis de vulnerabilidades con la herramienta Nessus de los servicios de la DMZ, de modo que se pueda hacer un estudio más profundo de su efectividad.

Llegados a este punto se establece una comparativa entre los tres y se analiza cuál de ellos es el más adecuado para según qué escenarios empresariales.

Todo ello no quita que este proyecto también pueda servir de ayuda para la elección de uno u otro *Firewall* en una red doméstica.

Title: ***Firewalls software*****: Study, setup, settings configuration and comparative**

Author: Mònica Ferrer Berbegal

Director: Marco Antonio Peña Basurto

Date: 14th January 2006

Overview

HomePC users and office users have a wide range of facilities to manage their work through the public Internet Network. However, this can bring to the problem of data violation in Network, if any security and control measures are not taken.

Major is the importance and the difficulty about data confidentiality in office environments to deal with it because the communication is established between the private company network and Internet. Therefore, this Project focuses in a company environment.

To guarantee a certain level of security between two or more Networks, Firewalls must be employed to prevent from undesirable communications.
This Project consists in the study, the implementation and the analysis of three Firewalls software in a Desmilitarizad Network, which contain common services used in the daily work of an enterprise. This services must be protected with a Firewall for a restricted access.

Firewalls are set up with applications of diferent Operating Systems**:**

- Iptables under Debian Sarge de GNU/Linux
- ISA Server 2004 Enterprise under Windows Server 2003.
- Packet Filter under OpenBSD.

The procedure followed has been:
Firstly, implementation and analysis of the advantages and disadvantages of each of them.

Secondly, with the aim of extracting a deeper study on effectivity, it is done a vulnerability services analysis with Nessus tool.

Finally, by the comparison and analysis of the three obtained results, conclude the work designing each Firewall its more efficient in office environments**.**

Notwithstanding, that the above referred study and conclusions of Firewalls analysis could also be applied to a Home networks.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este primer capítulo nos disponemos a explicar en qué consiste el proyecto para situar al lector, para ello dividimos el capítulo en diferentes apartados:

- Motivación
- Objetivos generales
- Planificación y análisis de costes
- Estructura de la memoria

En el apartado de Motivación describimos las motivaciones que nos mueven a hacer toda esta labor para alcanzar una serie de objetivos definidos en el siguiente apartado.

Un tercer apartado corresponde a la planificación temporal y de costes que se debe hacer previa al desarrollo del proyecto, la cual servirá de orientación.

Finalmente añadimos un cuarto apartado donde se explica cómo estructuraremos la memoria.

1.1 Motivación

Internet disfruta de una popularidad creciente entre los usuarios particulares y entre las empresas. El acceso directo y continuo a Internet está extendiéndose en los domicilios particulares y en el ámbito empresarial conforme los servicios de conexión ADSL se expanden en el mercado. Esto podría suponer un problema si no se tienen en cuenta una serie de medidas de seguridad.

Hoy en día la seguridad es un tema de estudio relevante y fundamental para poder preservar todo tipo de información y contenidos que requieran cierta privacidad. Gracias al continuo y rápido avance del mundo de la telemática, los usuarios disponen de un amplio abanico de comodidades que años atrás eran inconcebibles y ahora les permiten gestionar su trabajo a través de una pantalla y toda una red de ordenadores detrás de ella. Todo esto condicionado a ofrecer siempre una garantía de que en todo momento y bajo ningún concepto puedan ser manipulados los datos personales, ni tener acceso al sistema y mucho menos descubrir sus claves privadas. A sabiendas de que cada vez aumenta más el número de atacantes, que éstos están más organizados y que adquieren cada vez más habilidades, el tema de la seguridad adquiere mayor importancia.

Si la confidencialidad de nuestros datos ya es una preocupación para cualquier individuo, en una empresa esta cuestión adquiere mayor magnitud. Es por ello que mi proyecto se focaliza en un ámbito empresarial.

En este contexto, es obvia la importancia de encontrar soluciones al problema de la seguridad, así pues con el estudio de diferentes *Firewalls* pretendemos hacer una aportación en esta línea.

A lo largo de toda la carrera siempre he tenido especial interés en el campo de la seguridad de redes. Dentro del amplio mundo de la telemática es fundamental poder garantizar la seguridad en cualquier infraestructura de red a desarrollar; es por eso que he decidido escoger este título en mi proyecto final de carrera. Es un periodo de tiempo en el que yo como estudiante de Telecomunicaciones debo dedicar todo mi tiempo en el desarrollo de un tema que me motive lo suficiente para seguir ampliando los conocimientos adquiridos hasta el punto de presentar un estudio útil para cualquier empresa o cualquier persona que decida sacar provecho.

1.2 Objetivos generales

El objetivo de este proyecto es el estudio y la implementación de tres tipos de *Firewalls* bajo un escenario concreto, descrito con mayor detalle en el Capítulo 4. En base a este estudio concluiremos para qué entornos es más apropiado cada uno de los *Firewalls*.

Para configurarlos utilizaremos distintos sistemas operativos que nos proporcionan herramientas muy diversas:

Para empezar estudiaremos e implementaremos un *Firewall* bajo la distribución Debian Sarge de GNU/Linux como Sistema Operativo Open Source, cuya herramienta ofrecida, las *Iptables*, nos permitirá crear una serie de reglas que configurarán nuestro *Firewall*.

El segundo *Firewall* con el que trabajaremos es con la aplicación ISA Server 2004 Enterprise sobre Windows Server 2003.

Finalmente, de la misma manera que en los otros dos casos, configuraremos un tercer *Firewall* bajo el Sistema Operativo Open Source OpenBSD con la herramienta que nos proporciona *Packet Filter.*

En los tres casos se hará previamente un estudio teórico para familiarizarnos con los tres entornos y a continuación nos dispondremos a implementarlos en nuestras máquinas. Una vez configurados les someteremos a una batería de pruebas para verificar su correcto funcionamiento, ver sus limitaciones y vulnerabilidades con herramientas de escaneo. Teniendo unos objetivos más ambiciosos y queriendo ir más allá, nos dispondremos hacer una comparativa y estudiaremos lo que supondría la implementación del *Firewall* en distintos modelos empresariales.

Además, para poder tener una idea más consistente y profunda de las ventajas e inconvenientes de cada uno de los *Firewalls*, nos dispondremos a

configurarlos para otro tipo de escenario; según los servicios que se vayan a ofrecer y en que parte de la red se vayan a dar.

En definitiva, los objetivos marcados al comienzo de este proyecto se pueden resumir en:

- Configuración de los servicios que formarán parte del escenario:

  - DNS
  - SSH
  - Correo
  - Web
  - WINS
  - DHCP

- Estudio de la taxonomía de *Firewalls*.
- Análisis en profundidad a nivel teórico de los tres tipos de *Firewalls* , previo a su configuración:

  - *Iptables* sobre la distribución Debian Sarge de Linux.
  - ISA Server 2004 Enterprise sobre Windows 2003 Server
  - *Packet Filter* sobre la distribución OpenBSD

- Configuración de los tres *Firewalls* en base a una política de negación de servicio.

- Simulación de ataques y análisis de vulnerabilidades mediante el escaneo de puertos con la herramienta Nessus.

- Configuración de los *Firewalls* para distintos escenarios en función de la ubicación de los servicios.

  - Escenario con DMZ.
  - Escenario sin DMZ.
  - Escenarios en los que combinamos los servicios entre la DMZ y la Interna.
  - Valorar cada escenario y hacer una comparativa.

- Redacción de esta memoria escrita con objeto de documentar las conclusiones alcanzadas tras cumplir los anteriores objetivos.

1.3 Planificación y análisis de costes

Planificación temporal

Previo a la elaboración del proyecto debemos hacer una planificación temporal mediante el programa de planificación de proyectos Planner 0.13 (Véase Ilustración 1).

Esta planificación se ha creado con la máxima precisión posible pero siempre teniendo en cuenta ciertos márgenes para que la planificación teórica y la práctica difieran lo menos posible, pues siempre aparecen imprevistos que no se contemplan al empezar el proyecto y conllevan una duración más extensa.

WBS	Name	Start	Finish	Work	Duration	Slack
1	**Búsqueda y planificación del proyecto y de sus objectivos**	**mar 21**	**abr 29**	**30d**	**30d**	**51d**
1.1	Búsqueda y filtrado de información	mar 21	abr 29	30d	30d	51d
2	**Planificación junto con el tutor sobre el proyecto y cómo se desarrollará**	**ene 17**	**may 11**	**83d**	**83d**	**43d**
2.1	Planificación personal del trabajo	may 10	may 10	1d	1d	44d
2.2	Comentar planificació amb tutor i modificacions	may 11	may 11	1d	1d	43d
2.3	Creación de un índice	ene 17	ene 17	1d	1d	125d
3	**Estudio práctico del proyecto**	**may 13**	**may 24**	**8d**	**8d**	**34d**
3.1	**Instalación de diferentes sistemas operativos: Debian, OpenBSD y Windows 2003 Server**	**may 13**	**may 24**	**8d**	**8d**	**34d**
3.1.1	Instalación del Windows 2003 Server	may 13	may 13	1d	1d	41d
3.1.2	Familiarización e instalación de la distribución Debian	may 16	may 19	4d	4d	37d
3.1.3	Familiarización e instalación de OpenBSD	may 23	may 24	2d	2d	34d
4	**Presentación de los diferentes escenarios a implementar**	**may 23**	**may 23**	**1d**	**1d**	**35d**
4.1	"Comentarlos" con el tutor	may 23	may 23	1d	1d	35d
4.2	Redactar los escenarios en la memoria	may 23	may 23	1d	1d	35d
5	**Estudio de los escenarios con la distribución Debian (iptables)**	**may 24**	**jun 20**	**20d**	**20d**	**15d**
5.1	Ir transcribiendo en la memoria la configuración y las pruebas hechas	may 24	jun 20	20d	20d	15d
6	**Estudio de los escenarios de firewalls con ISA Server**	**jun 6**	**jun 24**	**15d**	**15d**	**11d**
6.1	Ir transcribiendo en la memoria la configuración y las pruebas hechas	jun 6	jun 24	15d	15d	11d
7	**Estudio de los escenarios de firewalls con OpenBSD (paquet filter)**	**jun 20**	**jul 1**	**10d**	**10d**	**6d**
7.1	Ir transcribiendo en la memoria la configuración y las pruebas hechas	jun 20	jul 1	10d	10d	6d
8	**Estudio comparativo de los tres tipos**	**jun 24**	**jul 6**	**9d**	**9d**	**3d**
8.1	Conclusiones con el escenario 1	jun 24	jun 24	1d	1d	11d
8.2	Conclusiones con el escenario 2 y 3	jun 24	jun 24	1d	1d	11d
8.3	Conclusiones con los posibles "escenarios adicionales"	jul 4	jul 6	3d	3d	3d
9	**Puesta en común entre tutor y alumno de las comparaciones de los escenarios**	**jun 27**	**jun 27**	**1d**	**1d**	**10d**
9.1	Modificaciones pertinentes	jun 27	jun 27	1d	1d	10d
10	Conclusiones del proyecto	jun 27	jun 27	1d	1d	10d
11	Lectura, Revisión de la memoria completa	jun 27	jun 29	3d	3d	8d
12	**Preparación de la presentación**	**jul 4**	**jul 5**	**2d**	**2d**	**4d**
12.1	Puesta en común con el tutor	jul 4	jul 4	1d	1d	5d
12.2	Modificaciones pertinentes	jul 5	jul 5	1d	1d	4d

Ilustración 1. Planificación temporal del Proyecto

Planificación de costes

Por lo que respecta al análisis de costes previstos para este proyecto adjuntamos una tabla Excel (Ilustración 2) en la que exponemos los gastos a distintos niveles.

Por una parte el material hardware que se usará, aplicando para cada elemento un precio aproximado.

Por otra parte adjuntamos el coste del software qué es lo que más encarece el proyecto dado que se utilizan aplicaciones de una entidad privada, Microsoft.

Finalmente, en base a la previsión temporal dada calculamos el coste de todas las horas invertidas en este proyecto; teniendo en cuenta que cada hora de un Ingeniero Técnico de Telecomunicaciones recién titulado, como es mi caso, se cobra a 15 €. Las fuentes vienen dadas por la experiencia de compañeros que actualmente están trabajando en colaboración de proyectos.

Todo ello resulta dar un coste aproximado del proyecto de **20.230,71 €**.

Item	Descripción	Unidades	Euros por unidad	Euros totales
HARDWARE				
Hardware	Pentium IV clónico	4	850,00 €	3.400,00 €
	Switch 8 puertos	3	20,00 €	60,00 €
	Cable Ethernet 10Base-T. Par trenzado UTP cat 5. (unidad metro)	10	0,45 €	4,50 €
	Conector RJ-45	20	0,30 €	6,00 €
	Cd´s vírgenes	10	0,60 €	5,38 €
	Total			3.475,88 €
SOFTWARE				
Software	Windows 2003 Server con licencia	1	841,51 €	841,51 €
	ISA Server 2004 Enterprise	1	5.053,32 €	5.053,32 €
	Total			5.894,83 €
MANO DE OBRA				
Ingeniero Técnico	Euros por hora	724	15,00 €	10860,00 €
COSTE TOTAL				
Coste total proyecto	Proyecto global			**20230,71** €

Tabla 1. Análisis de costes previstos

1.4 Estructura de la memoria

La memoria del proyecto se estructura en varios capítulos que se distribuyen de la siguiente manera:

Es fundamental en cualquier proyecto, situar al lector aún sin haberse leído la memoria entera y para ponerle en contexto; por ello hacemos una introducción. En el apartado de Introducción (Capítulo 1) se explica el porqué del proyecto, las motivaciones que nos mueven a hacer toda esta labor y los objetivos a los que se quieren llegar. Además se debe dar unas pautas temporales y económicas sobre lo que va a costar llevar este proyecto a priori. Una vez leída esta parte el lector debe tener ya una idea concreta y bien formada del trabajo sin tener que leerse todo su contenido.

A continuación, el Capítulo 0 se refiere con mayor detenimiento lo que es un *Firewall*, su taxonomía y las políticas de seguridad por las que se rigen. En este apartado lo que se pretende es facilitar la comprensión del concepto de *Firewall*, clave para el entendimiento de todo el proyecto.

Una vez explicado de manera teórica el concepto de la temática principal del proyecto, describimos en un tercer Capítulo (Ataques) los diferentes modos de detectar un ataque, además de explicar algunos de los que utilizaremos en la práctica para comprobar el funcionamiento del *Firewall* que configuramos.

A continuación, en el Capítulo 4 (Escenario) detallamos la configuración del escenario en el que nos encontramos; pues seguirá la misma estructura de una de las arquitecturas que ya habremos explicado en el Capítulo 2. De esta manera el lector puede ir entrelazando todo aquello que va leyendo e irse ubicando en todo el contexto del proyecto.

Este capítulo consta de varios puntos:

- Esquemas ilustrativos
- Descripción del escenario
- Configuración de la red
- Servidores

En el primer apartado mostramos el escenario en modo esquemático para un mayor entendimiento. A continuación explicamos las diferentes redes y elementos que forman parte del escenario. Un tercer apartado para desarrollar la configuración de la red siendo que intervienen tres subredes distintas.
Finalmente en el apartado de Servidores describimos todos los servicios que va a disponer nuestra empresa y la configuración pertinente.

Seguidamente, encontramos en tres capítulos distintos (Capítulo 5, 6 y 7) implementación de los tres *Firewalls* (*Iptables*, ISA Server 2004 Enterprise y *Packet Filter*) respectivamente. Todos ellos siguen una misma estructura que consiste en los siguientes puntos:

- Descripción del *Firewall*
- Configuración del *Firewall*
- Análisis de vulnerabilidades
- Valoración del *Firewall*

En este punto estudiamos más a fondo de cada uno de los *Firewalls*, puesto que teniendo el conocimiento teórico y la experiencia práctica podremos extrapolar conclusiones que describiremos en el punto de Valoración del *Firewall*.

A continuación nos disponemos a detallar los puntos en base a los cuales hacemos la valoración del *Firewall* ya que son varios los factores que nos ayudan a hacer esta valoración.

- Sistema Operativo: S.O sobre el que se trabaja.
- Política de seguridad: Tenemos que tener en cuenta cual es la política por defecto del *Firewall* para modificarla si es necesario.
- Configuración de las reglas: Describimos si la configuración es sencilla o compleja.
- Tipo de filtrado: Cada *Firewall* se identifica por el nivel de filtrado que utiliza.
- Calidad de Servicio: Determinamos cómo es capaz el *Firewall* de realizar si es posible priorizar de tráfico.
- Características: Describimos más características encontradas en base al estudio teórico y práctico.
- A modo de conclusión: Como resumen conclusivo.

Para terminar el Capítulo 8. se centra en hacer un balance sobre las conclusiones en base al estudio práctico. Éste consta de dos apartados:

- Análisis comparativo de los tres *Firewalls*
- Adecuación de los *Firewalls* según el escenario empresarial

En el primero extrapolamos las conclusiones en base a las valoraciones descritas de cada uno de los *Firewalls*. Mientras que en el segundo apartado presentamos posibles escenarios empresariales y hacemos un estudio sobre cual de los *Firewalls* sería el más adecuado.

Seguidamente, en el Capítulo 9. se resumen las conclusiones alcanzadas en el proyecto, se revisa la planificación temporal y los costes y se apuntan algunas líneas futuras de investigación relacionadas con el estudio de estos *Firewalls*.

Por último, en los anexos del proyecto se recogen los ficheros de configuración creados y las tablas de los análisis de vulnerabilidades.

2. *FIREWALLS*

En este capítulo explicamos con mayor detenimiento lo que es un *Firewall*, su taxonomía y las políticas de seguridad por las que se rigen.
El capítulo consta de varios apartados:

- ¿Qué es un *Firewall*?
- Arquitecturas de *Firewalls*
- Políticas de seguridad

2.1 ¿Qué es un *Firewall*?

Un *Firewall* o Cortafuegos es un sistema de seguridad de redes en el que se protege una máquina o subred de servicios que desde el exterior puedan suponer una amenaza a su seguridad. Dicho en otras palabras, es un sistema de aislamiento entre dos o más redes para evitar comunicaciones indeseadas.

El *Firewall* actúa de filtro, de manera que examina todos y cada uno de los paquetes de información en base a unas reglas definidas. Es bajo una política de seguridad que decide qué paquetes puede aceptar, cuáles modificar o cuáles bloquear.
Según como realice el filtrado, un *Firewall* puede configurarse de varios modos:

- *Firewall* de filtrado de paquetes: Normalmente es un router que selecciona según las IPs y los puertos de salida. Éstos *Firewalls* trabajan a nivel de red.
- Servidores Proxy: Trabajan a nivel de aplicación. Hay de dos tipos:
    - Proxy Caché. Transparentes + aplicación. Se mira a nivel de aplicación.
    - Proxy Socks: Filtrado a nivel de TCP (transporte).

Un *Firewall* no tiene porque ser una sola máquina sino que se compone de varios elementos:

- *Bastion host*. Máquina que debemos proteger. Se realiza el filtrado a nivel de aplicación y normalmente se conecta en la zona interior con la zona exterior. Es aquí donde se sitúan los servicios.
- *Choke*. Filtro de paquetes.
- Proxy

Uno de los inconvenientes de los *Firewalls* es que no aíslan de los ataques de dentro de la misma red, por ello es conveniente utilizar *Firewalls* internos. Por ejemplo, para una gran empresa puede haber diferentes *Firewalls* para cada departamento y así hay mayor seguridad ante un daño generalizado.

En cuanto a la taxonomía de *Firewalls*, existen dos grandes tipos de *Firewall*:

- *Firewall* de software
- *Appliance*

2.1.1 *Firewall* de *Software*

Un *Firewall* de software es una aplicación que puede estar integrada en el mismo sistema operativo (*Iptables* y *Packet Filter*) o puede instalarse independientemente (ISA Server).

Estos *Firewalls* se configurarán en base a la política de acceso que desee el usuario y su implementación dependerá de la dificultad que tenga el propio *Firewall*, de la velocidad de la máquina, de la memoria, etc.

Las actualizaciones de los *bugs* de seguridad que van apareciendo deberán ser configuradas manualmente, que en según qué casos no es una tarea sencilla.

A parte de estos *Firewalls* mencionados, existen los llamados *Firewalls* personales que son mucho más sencillos para uso doméstico.

2.1.1.1 Firewalls personales

En las redes personales o conexiones permanentes domésticas, lo habitual es que el ordenador esté conectado directamente a Internet, de ahí la necesidad de un *Firewall* por software.

Éstos *Firewalls* no son tan complejos como otras soluciones profesionales pensadas para empresas _los *Firewalls* que analizaremos en el proyecto_ pero en general cumplen bien con su cometido y son suficientes para la seguridad de un ordenador conectado a Internet o una red doméstica.

Siendo que son unos *Firewalls* más limitados, su funcionalidad se basa en bloquear *popup's*.

2.1.2 Appliances o "*Firewalls* físicos"

Un *Firewall* físic*o* o *appliance* es un sistema hardware y software cuya única función es la de implementar una política de acceso. A diferencia de los *Firewalls* de Software estos suelen estar ya preconfigurados, de tal manera que sólo sea necesario conectarlos en la red. A parte, a diferencia de los *Firewalls* de software las actualizaciones de *bugs* son automáticas.

Un *appliance* bien equipado debe incluir la implementación NAT (*Network Address Translation*), DMZ (*De-Militarized Zone*), VPN (*Virtual Private Network*), detección de intrusos y un programa de auditoría con alarmas.

2.2 Arquitecturas de *Firewalls*

Existen tres grandes tipos de *Firewalls*:

- *Firewalls* de la capa de red
  - Screened Host
  - Screened Subnet
- *Firewalls* de la capa de aplicación
  - Dual-Homed Host
- *Firewalls* híbridos

Éstos *Firewalls* tienen claras diferencias lo que no significa que uno sea mejor que otro, puesto que cada uno de ellos es apropiado para un entorno específico. Lo que los diferencia unos de otros es el mecanismo que utiliza para dejar pasar el tráfico de una zona a otra. El modelo de ***I**nternational **S**tandards **O**rganization* (ISO) ***O**pen **S**ystems **I**nterconnect* (OSI) define siete capas, donde cada una de ellas proporciona los servicios que capas superiores requieren de ellos.

Ilustración 2. Modelo de capas de la torre OSI

Es importante tener en cuenta que cuanto más bajo sea el nivel al que pertenece el mecanismo de *forwarding*, menos sofisticada será la inspección del paquete, con lo cual estos *Firewalls* serán más rápidos pero más sensibles ante una amenaza.

Hoy en día existen los *Firewalls* híbridos que hacen el filtrado por red y algunas inspecciones a nivel de aplicación cuya examinación del paquete dependerá del vendedor, del producto, del protocolo y la versión.

2.2.1 Screened Host

Este tipo de cortafuegos pertenece al grupo perteneciente a la capa de red, de manera que filtra por direcciones origen y destino, por protocolo o por puertos origen y destino; es decir parámetros de la capa TCP/IP.

La arquitectura con la que se forma combina un *router* con un *host* bastión donde el principal nivel de seguridad proviene del filtrado de paquetes (*Packet Filter*), es decir, el *router* es la primera y más importante línea de defensa. En la máquina bastión, único sistema accesible desde el exterior, se ejecutan los *proxies* de las aplicaciones, mientras que el *choke* se encarga de filtrar los paquetes que se puedan considerar peligrosos para la seguridad de la red interna, permitiendo únicamente la comunicación con un número reducido de servicios.

Esta arquitectura está cada vez más en desuso debido a que presenta dos puntos únicos de fallo, el *choke* y el bastión: si un atacante consigue controlar cualquiera de ellos, tiene acceso a toda la red protegida; por tanto, es más recomendable una arquitectura de red perimetral, de la que vamos a hablar a continuación.

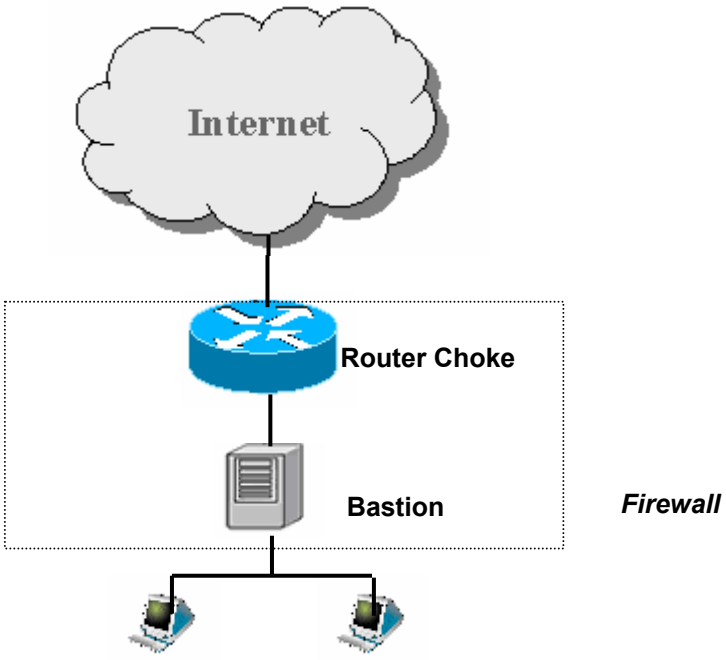


Ilustración 3. Arquitectura Screened Host

2.2.2 Screened Subset o DMZ

De la misma manera que la arquitectura anterior Screened Subnet red perimétrica pertenece al grupo de *Firewalls* que filtran según el nivel de red.
La red perimétrica también conocida como *De-Militarized Zone* o *Screened Subnet*, es la más utilizada e implantada hoy en día. Ésta añade un nivel de seguridad en las arquitecturas de cortafuegos situando una subred (DMZ) entre las redes externa e interna, de forma que se consiguen reducir los efectos de un ataque exitoso: como hemos venido comentando, en los modelos anteriores toda la seguridad se centraba en el bastión, de forma que si la seguridad del

mismo se veía comprometida, la amenaza se extendía automáticamente al resto de la red. Como la máquina bastión es un objetivo interesante para muchos piratas, la arquitectura DMZ intenta aislarla en una red perimétrica de forma que un intruso que accede a esta máquina no consiga un acceso total a la subred protegida. Este tipo de arquitectura es con la que nos disponemos a trabajar, pues la seguridad se ve aumentada si se diseñan redes *perimétrica*s o DMZ protegidas por *Firewalls*.

Con este tipo de arquitectura los servicios de acceso al público y por lo tanto los servidores donde están configurados se encuentran protegidos por un *Firewall* externo o *bastion* y la red interna por un *Firewall* interno o de contención.

Si ponemos los servidores en la red interna, sabiendo que pueden acceder usuarios de la red externa es mucho más fácil que haya un agujero que permita al usuario de la red externa entrar en el sistema de la red interna. Pues lo que sí puede suceder es que haya un acceso no autorizado a uno de los servidores localizados en la DMZ.

A parte podría ocurrir que un intruso coloque una herramienta de monitorización de red o *sniffer*, lo cual provocaría que los paquetes IP que pasen de la red DMZ a la red interna fueran capturados y analizados, permitiendo al intruso conseguir información para poder hacer un ataque a la red interna.

Este modelo se puede diseñar en dos modos distintos:

- Dos *Firewalls*, cada uno con dos adaptadores de red instalados. El *Firewall* externo que comunica la DMZ y uno interno que comunica la DMZ con la red interna.

- Un sólo *Firewall*, con tres adaptadores de red (nuestro caso); con el que se comunica la red externa, la zona desmilitarizada y la red interna.

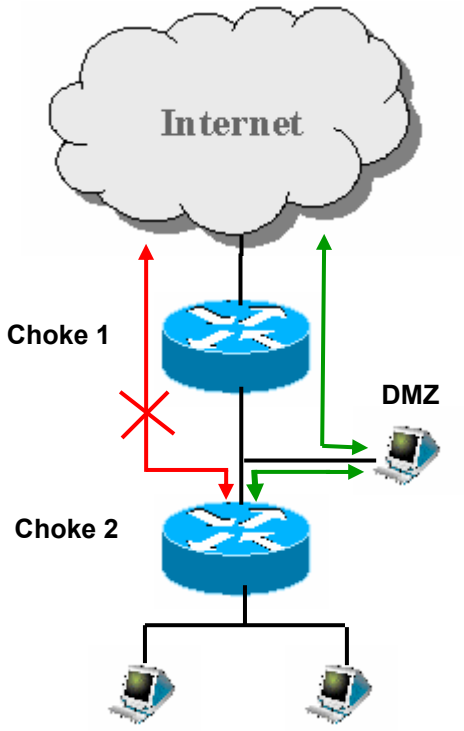


Ilustración 4. Arquitectura de red perimetral (DMZ)

2.2.3 Dual-Homed Host

Este modelo de *Firewall* perteneciente al grupo de filtrado a nivel de aplicación está formado por máquinas equipadas con dos o más tarjetas de red y en las que una de las tarjetas se suele conectar a la red interna a proteger y la otra a la red externa a la organización. En esta configuración el *choke* y el bastión coinciden en el mismo equipo.

El sistema debe ejecutar al menos un servidor *proxy* para cada uno de los servicios que deseemos pasar a través del *Firewall* y también es necesario que el *IP Forwarding* esté deshabilitado en el equipo, pues aunque una máquina con dos tarjetas puede actuar como un *router* para aislar el tráfico entre la red interna y la externa es necesario que el *choke* no enrute paquetes entre ellas. Así, los sistemas externos `verán' al *host* a través de una de las tarjetas y los internos a través de la otra, pero entre las dos partes no puede existir ningún tipo de tráfico que no pase por el *Firewall.*

Por otra parte, el hecho de que haya un Proxy existirá una aplicación de software que permitirá un amplio número de procesos de carga y de control de acceso, lo que obliga un filtrado mucho más elaborado y consecuentemente un mayor tiempo de carga.

El tráfico cruzará de una red a la otra después de un “enmascaramiento” efectivo de la conexión inicial. Los *Firewalls* a nivel de aplicación permiten un refuerzo en la seguridad debido a una auditoría mucho más detallada que los *Firewalls* de nivel de red.

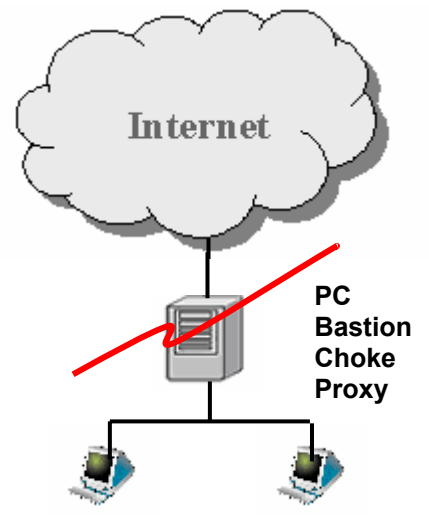


Ilustración 5. Arquitectura Dual-Homed

2.2.3.1 Proxy Server

Un Proxy Server (a veces referido a una aplicación *gateway* o de reenvío) es una aplicación que controla el tráfico entre una red protegida e Internet. Muy a menudo, los proxies sustituyen los routers de control de tráfico para prevenir el tráfico que pasa directamente entre las dos redes. Además muchos proxies contienen soporte adicional para la autenticación de usuarios.

Desde que los proxies tienen la capacidad de controlar el protocolo de aplicación que se utiliza pueden implementar una seguridad específica para ese protocolo (por ejemplo un Proxy FTP debe estar configurado para permitir el tráfico FTP de entrada y bloquear el de salida).

2.3 Políticas de seguridad

2.3.1 ¿Qué se entiende por política de seguridad?

Todo *Firewall*, sea del tipo que sea, se rige por una política de seguridad definida previamente a su configuración. Cuando hablamos de política de seguridad nos referimos a tener una directiva predeterminada y una colección de acciones a realizar en respuesta a tipos de mensajes específicos. Cada paquete se compara, uno a uno, con cada regla de la lista hasta que se

encuentra una coincidencia. Si el paquete no coincide con ninguna regla, fracasa y se aplica la directiva predeterminada al paquete.

Hay dos perspectivas básicas o políticas de seguridad para un *Firewall*:

- Denegar todo de forma predeterminada y permitir que pasen paquetes seleccionados de forma explícita.
- Aceptar todo de forma predeterminada y denegar que pasen paquetes seleccionados de forma explícita.

La directiva de denegar todo es la propuesta más segura y por lo tanto con la que vamos a trabajar; pues facilita la configuración de un *Firewall* seguro pero es necesario habilitar cada servicio sabiendo el protocolo de comunicación para cada servicio que se habilite.

La directiva de aceptar todo facilita mucho la configuración y la puesta en funcionamiento de un *Firewall* pero obliga a prever todo tipo de acceso imaginable que se quiera deshabilitar. El peligro es que no se preverá un tipo de acceso peligroso hasta que sea demasiado tarde, o posteriormente habilitará un servicio no seguro sin bloquear primero el acceso externo al mismo. En definitiva, programar un *Firewall* seguro para aceptar todo, implica más trabajo, mayor dificultad y por tanto es más propenso a errores.

Por otra parte, cabe decir que es muy importante para la gestión de un *Firewall* el menor número de reglas posibles.

En base a esto, nos disponemos a definir cuál es nuestra política de seguridad y los permisos de entrada o salida de cada una de las subredes de nuestra red.

2.3.2 Política de seguridad utilizada

Como hemos explicado en el punto anterior, nuestros *Firewalls* van a seguir la política de denegar todos los paquetes exceptuando todo aquello que tengan una entrada o salida habilitada mediante una o varias reglas.

De manera gráfica, podemos ver lo que ocurre con este tipo de directiva (véase Anexo I-A).

Es importante que de partida se tenga claro los servicios que van a ser habilitados para cada una de las subredes, de manera que luego creemos las correspondientes reglas y todo lo que no coincida con ello será descartado.

Pudiendo ver el esquema de nuestra red en el Capítulo 4. podemos definir las siguientes pautas:

Servicios accesibles desde la red Externa a la DMZ y viceversa

- Externa→DMZ

Todo cliente podrá ver los servicios que ofrece la empresa a través de la llamada "zona neutral" en la que se colgarán los servicios Web, Secure Shell, Correo y DNS que se explicarán con mayor detalle en el siguiente punto.

Entonces tenemos que habilitar el tráfico desde cualquiera de las redes externas (0.0.0.0) a la red de la DMZ (192.168.200.0/24) para paquetes cuyos protocolos sean HTTP, HTTPS, SMTP y DNS.

No habilitamos el servicio SSH desde la red externa ya que consideramos que sólo es admisible para personal autorizado conectado desde el *Firewall* o la propia red de la empresa.

- DMZ→Externa

En sentido inverso no va a ser necesario habilitar tráfico puesto que en la DMZ simplemente se van a ubicar los servicios y ninguna otra máquina de usuario.

Servicios accesibles desde la red Interna a la DMZ y viceversa

- Interna→DMZ

En este punto se habilitan los paquetes de la misma manera que en el caso anterior, dando acceso a los servicios que nos ofrece la DMZ (WEB, MAIL, SSH y DNS).

- DMZ→Interna

Hacemos especial hincapié en que se va a descartar cualquier paquete que vaya en este sentido puesto que no es imprescindible permitir el paso de ningún tipo de tráfico y lo único que provocaría es que existiera un agujero para llegar desde la red externa a la red interna.

Además, cabe decir que el sentido de poner en este punto algún servicio es para habilitar única y exclusivamente el acceso a los mismos trabajadores de la empresa, de ahí que no tengamos la necesidad de habilitarlos ninguna regla referida al tráfico de entrada de la red interna.

Servicios accesibles desde la red Interna a la Externa y viceversa

- Interna→Externa

Los trabajadores de la red interna podrán tener acceso a los servicios más comunes de (DNS, SMTP, POP3, SSH, TELNET y WEB), el resto, quedarán cerrados. Con ello nos aseguramos que los programas cuyos protocolos son Peer to Peer, usados muy habitualmente hoy en día, queden deshabilitados;

Estas aplicaciones serían un factor añadido de inseguridad que puede ser evitado.

- Externa→Interna

Obviamente se va a deshabilitar todo tipo de tráfico porque sino perdería el sentido el colocar una red desmilitarizada que proteja la red interna.

Para verlo de modo esquemático hacemos referencia en el Anexo I-B a un cuadro resumen en el que aparecen los servicios asociados a sus respectivos puertos que habilitará o deshabilitará el *Firewall* según la dirección del tráfico.

3. ATAQUES

En este capítulo nos disponemos a describir los diferentes tipos de ataques, además de explicar con mayor detalle el funcionamiento de la herramienta Nessus como técnica utilizada para la detección de vulnerabilidades.

La estructura que sigue es la siguiente:

- Antes de una ataque
- Escaneo de puertos
- Negaciones de servicio
- *Spoofing*
- *Sniffing*
- Herramientas de escaneo de vulnerabilidades

3.1 Antes de un ataque

Antes de empezar un ataque cualquier asaltante debe conocer, con la mayor precisión posible, a qué se enfrenta, es por eso que en el primer ataque se querrá obtener la información de la versión del servicio. Debemos evitar proporcionar información gratuita, ya que podría ser utilizada de manera maliciosa por parte de un individuo con intención de asaltar.

La información fundamental que necesita un atacante es la dirección IP de la víctima, o de lo contrario el nombre del dominio que pretende atacar. Con un simple paquete ***ICMP*** es capaz de obtener la dirección IP de cualquier dominio lo cual debemos evitarlo en la configuración de nuestros *Firewalls*.

Por otro lado, es importante distinguir los ataques que se realizan desde Internet, accediendo a nuestro servidor a través de la dirección IP pública y los que se realizan desde dentro de nuestra propia red. En el caso de los ataques internos, si éstos los hace algún usuario de la empresa, se acentúa la peligrosidad de estos debido a que poseen más información sensible sobre nuestra red.

A continuación describiremos algunos de los ataques más comunes y el modo de detectarlos.

3.2 Escaneo de puertos

Una de las primeras actividades que un atacante realizará contra su objetivo será sin duda un escaneo de puertos; esto le permitirá obtener en primer lugar información básica acerca de qué servicios estamos ofreciendo en nuestras

máquinas y, adicionalmente, otros detalles de nuestro entorno como el sistema operativo del *host* o ciertas características de la arquitectura de la red. Analizando qué puertos están abiertos en un sistema, el atacante puede buscar agujeros en cada uno de los servicios ofrecidos: cada puerto abierto en una máquina es una potencial de puerta de entrada a la misma.

Para ello, podemos utilizar la aplicación TELNET. Si por ejemplo intentamos conectarnos a la DMZ y comprobar si el puerto 21 responde:

```
telnet 192.168.200.2 21
```

De este modo podemos saber la versión del Servicio que hay detrás (en este caso FTP).

Ilustración 6. Telnet al puerto FTP

Si responde significa que está abierto y escuchando peticiones. Si por lo contrario hay un *Firewall* protegiendo el puerto no obtendremos respuesta alguna.

Por otra parte, existe la posibilidad de hacer un escaneo masivo con herramientas como Nmap o Strobe que implementan diferentes técnicas para detectar también la versión del sistema operativo usado en la máquina atacada pasando inadvertidos.

3.3 Negaciones de Servicio (DoS)

Las negaciones de servicio o *Denial of Service* son ataques dirigidos contra un recurso informático con el objetivo de degradar total o parcialmente los servicios prestados por ese recurso. En entornos donde la disponibilidad es valorada por encima de otros parámetros de la seguridad global puede convertirse en un serio problema, ya que se podría interrumpir constantemente un servicio sin necesidad de grandes recursos. Por otra parte este tipo de ataques constituyen en muchos casos uno de los ataques más sencillos y contundentes contra todo tipo de servicios.

La inhabilitación de un servicio se suele hacer porque el atacante lo ha bloqueado totalmente o porque el propio servicio ha estado sometido a constantes ataques DoS.

3.3.1 Barridos PING

Uno de los ataques más comunes de Denegación de Servicio es el envío continuado de paquetes ICMP contra una dirección de *broadcast*, de manera que respondan todas las máquinas que se encuentren en esa red.

Este tipo de ataque puede ser fácilmente detectado por el administrador de la red ya que genera una gran cantidad de tráfico.

3.4 *Spoofing*

El *spoofing* o engaño de una dirección IP consiste en suplantar la identidad de una máquina. El intruso simula la identidad de otra máquina de la red para conseguir acceso a recursos de un tercer sistema que ha establecido algún tipo de confianza basada en el nombre o la dirección IP del *host* suplantado.

El propósito de estos ataques es tentar al host atacado, para que acepte datos como si vinieran de la fuente original o bien para recibir datos que deberían ir hacia la máquina suplantada y alterarlos.

3.5 *Sniffing*

El *Sniffing* ("husmear en la red") es un tipo de ataque de interceptación en el que una máquina diferente a la máquina destino lee los datos de la red. Esto tiene que ver con el uso de una interfaz de red para recibir datos no destinados a la máquina donde se encuentra dicha interfaz.

Las interfaces de red (*NIC - Network Interface Card*) pueden funcionar en modo promiscuo, en el cual la tarjeta lee todas las tramas que circulan por la red, tanto dirigidas a ella como a otras máquinas; el leerlas no implica el eliminarlas de la red, por lo que el *host* destino legítimo la recibirá y eliminará sin notar nada extraño.

3.6 Herramienta de escaneo de vulnerabilidades

Una vez vistos los diferentes ataques a los que toda máquina está sometida en la red, describiremos la herramienta con la que trabajaremos para analizar las vulnerabilidades de los *Firewalls* a configurar.

Pues en este punto el papel del *Firewall es* fundamental para ver qué puertos son los que se encuentran abiertos y cuáles no y cómo éste responde a los distintos ataques que genera esta herramienta, para cada uno de los escenarios de los tres *Firewalls* van a ser exactamente iguales.

Los ataques que vamos a realizar van a ser contra la red perimetral, concretamente con la máquina donde se ubican todos los servidores. El ataque lo haremos a la IP pública de la DMZ porque es ésta la que conocerá el atacante (192.168.16.214). De este modo podremos ver no sólo los puertos que están o no habilitados sino también las vulnerabilidades de los servicios que hay detrás.

Para hacer todas estas pruebas utilizaremos la herramienta Nessus 2.2.4 sobre un sistema operativo GNU/Linux Knoppix 3.9.

3.6.1 Configuración de Nessus

Nessus es uno de los auditores de seguridad más potentes ya que es una herramienta modular y extensible con mucha versatilidad.

Esta aplicación no es tan sólo un escaneador de puertos sino que también detecta las vulnerabilidades que hay en los servicios detrás del *Firewall*. Entre otras muchas cosas, esta herramienta trata de abrir los puertos y busca qué versión del servicio está funcionando, para luego ofrecer recomendaciones para atajar cualquier vulnerabilidad que pueda tener el servicio.

Para configurarlo debemos primero crear un usuario: nessus-adduser con su login y password y a continuación activamos el *daemon*.

```
/usr/sbin/nessus start
```

Posteriormente se configura la parte cliente indicando el servidor (el propio localhost) y el puerto al que se va a conectar #1241; y nos autenticamos.

Una vez se ha establecido la sesión se puede empezar a configurar las características de las pruebas que se van a llevar a cabo.

3.6.2 Parámetros de las pruebas

Para efectuar los ataques con la herramienta Nessus, hemos habilitado todos los *plugins* disponibles (FTP. CISCO, DoS, *Firewalls*, General, SNMP, Windows, etc).

Como opciones de escaneo hemos designado un rango de puertos amplio que englobe todos los servicios más comunes (1-8000) pues las herramientas que utiliza Nessus para el escaneo son las siguientes:

- *Scan for LaBrea tarpitted hosts*

Esta herramienta detiene los ataques de gusanos mediante una “trampa virtual” impidiendo que se extienda. Los servidores que tengan instalado este software simulan la presencia de una máquina que posee las características propias de un sistema vulnerable. Aunque realmente no existe ésta máquina puede ser detectada por escaneadores de puertos, sniffers y otras herramientas de

hacking. Cuando un hacker intenta conectarse a ella, esta ”trampa virtual” bloquea su conexión.

- Nessus TCP Scanner

Realiza un análisis de los puertos del *host*, una vez se establece la conexión obtiene los *banners* de los servicios para los *plugins* de identificación de servicios.

- SYN Scan

Se realiza un análisis de las direcciones IP destino teniendo en cuenta el RTT (*Round Trip Time*), que será el tiempo total de ida y vuelta de los paquetes que lanza el host de Nessus hasta el *host* remoto.

- *Exclude toplevel domain wildcard host*

Verificamos que se lanza Nessus contra un dominio de primer nivel.

- Nmap (NASL Wrapper).

  Es la herramienta Nmap en formato *Nessus Attack Scripting Language.* Las opciones de esta herramienta son varias y las que rigen nuestros chequeos son las siguientes:

  - *Xmas Bree Sian* que envía los paquetes con los flan FIN, URG, y PUSH activados.
  - UDP port scan
  - Service scan
  - RPC port scan
  - *Identify the remote OS*
  - *Fragmented IP packets (bypasses firewalls)*
  - *Get identd info*

Por otra parte todas nuestras pruebas estarán sometidas a un escaneo AGRESIVO.

- Netstat ‘scanner’

Con este *plugin* Nessus hará un Netstat para ver los puertos que tiene abiertos el sistema.

En nuestro caso no hemos habilitado ni el SYN Scan ni el Nessus TCP Scanner porque la funcionalidad de estos la incluye el *plugin* de Nmap.

3.6.3 Estructura del informe de vulnerabilidades

Nessus nos ofrece los informes en varios formatos, siendo que el más completo es en formato HTML más gráficos más gráficos es con el que vamos a trabajar.

Para estudiar las vulnerabilidades de cada *Firewall* seguiremos las mismas pautas que sigue el informe de Nessus.

En primer lugar adjuntaremos una tabla a modo resumen de los servicios que Nessus detecta en la red junto con el tipo de mensaje que tiene sobre vulnerabilidad para cada uno de ellos. Tabla de servicios del *host*

A continuación añadiremos las estadísticas de seguridad extraídas por la aplicación en modo gráfico, pues incluiremos tres gráficas:

- Una gráfica de barras en la que se muestra el tráfico que hay en la red del host, que viene dada a través del escaneo de puertos Nmap comprobando así cuáles son los que tiene habilitados la máquina a través del *Firewall*. Gráfica del tráfico en la red

- A continuación podemos ver una gráfica de disco (Gráfica de los riesgos de seguridad) en la que Nessus nos muestra el porcentaje de información de riesgo a través de una escala de tres niveles:
    - Bajo
    - Medio
    - Alto

- En el caso de que haya un porcentaje de alto riesgo provocado por algún agujero de seguridad, Nessus crea una gráfica en la que muestra qué servicio o servicios generan el problema de seguridad. Gráfica de *bugs*

Finalmente, Nessus genera una tabla en la que describe las vulnerabilidades de cada servicio según dos tipos de mensajes:

- *Bugs* o Agujeros de seguridad. *Security holes* (riesgo de seguridad alto)
- Avisos de seguridad. *Security Warnings* (riesgo de seguridad medio)
- Notificaciones de seguridad (riesgo de seguridad bajo)

Esta misma la adjuntaremos en el Anexo III además, de analizarla en los *Firewalls* pertinentes.

4. ESCENARIO

En este capítulo definimos la configuración del escenario en el que nos encontramos; por ello describimos cada una de las máquinas y dispositivos que forman parte del escenario, el software que disponen y los servicios que implementamos en cada una de las redes, junto con la configuración pertinente. De esta manera, la estructura del capítulo es la siguiente:

- Esquemas ilustrativos
- Descripción del escenario
- Configuración de la red
- Servidores

4.1 Esquemas ilustrativos

Para un mejor entendimiento, nos disponemos a ver en modo gráfico el escenario en el que vamos a realizar todas las pruebas.

4.1.1 Esquema funcional

Como hemos explicado en el anteriormente, vamos a diseñar una red perimetral, ya definida en el apartado 2.2.2.

De manera ilustrativa, nuestro escenario sigue el siguiente esquema

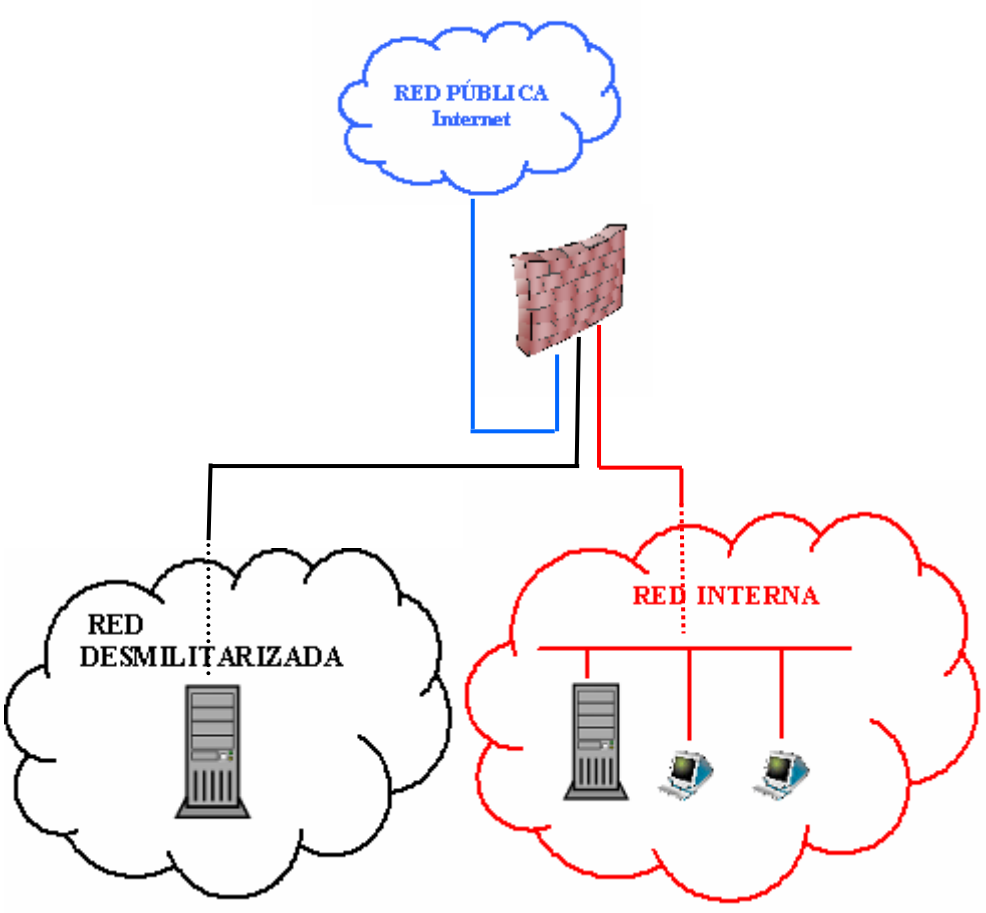


Ilustración 7. Esquema funcional del escenario

4.1.2 Esquema de direcciones

El esquema de direccionamiento IP y las interfaces de cada una de las máquinas es el siguiente:

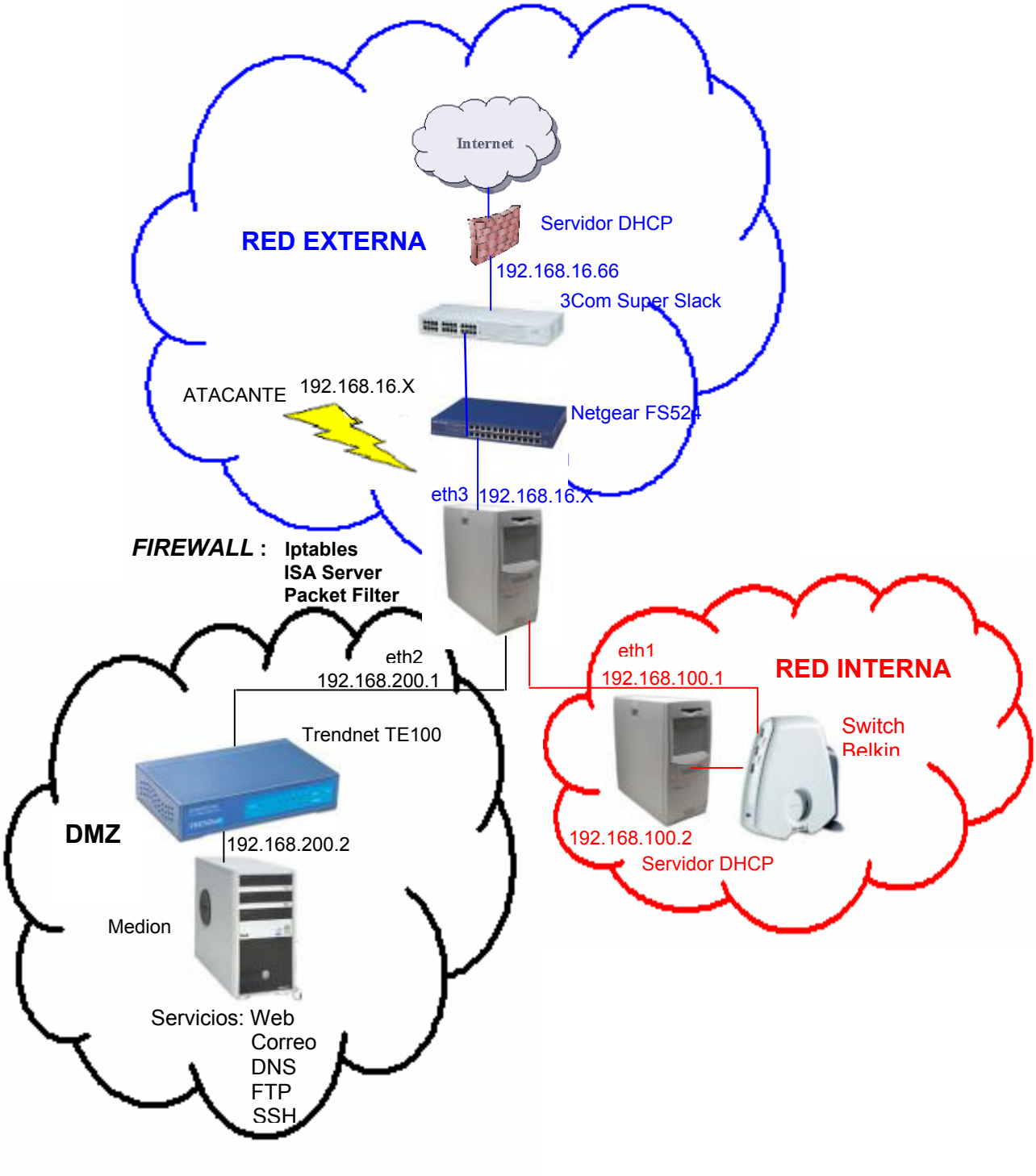


Ilustración 8. Esquema de direcciones del escenario

4.2 Descripción del escenario

Como podemos observar en las dos ilustraciones y siguiendo el esquema clásico de una arquitectura perimetral, nuestro escenario consta de varias subredes:

Red Externa (192.168.16.0/24)

Es la que nos conecta directamente a la red pública de Internet atravesando la una máquina que hace de *gateway* cuya IP es 192.168.16.66.

A parte, como podemos ver en la Ilustración 8. es en esta misma red desde la que se conectará el intruso para hacer los ataques a nuestra red interna y a la DMZ; pues simulamos que es "la red pública" a pesar de su direccionamiento privado.

Red Interna (192.168.100.0/24)
La interfaz 1 del *Firewall* (192.168.100.1) es la entrada a la red interna, la cual está formada por una sola máquina, la de Medion 192.168.100.2, que actuará también como servidor de DHCP. El motivo por el que ubicamos este servidor en este punto es porque lo que se quiere es poder tener un servidor que ofrezca única y exclusivamente IPs a máquinas de la red interna, sin tener la necesidad de que alguien de fuera pueda acceder a este servidor.

Ambas máquinas se conectan a través del *switch* Belkin, así podríamos conectar más máquinas que formen la LAN privada, las cuales obtendrán una dirección IP de manera dinámica.

Red Desmilitarizada o Perimetral (192.168.200.0/24)
Una tercera subred será la que la forme la máquina de la DMZ junto con el switch Trendnet. De esta manera en un futuro siempre se pueden conectar más máquinas que ofrezcan otro tipo de servicios.

La DMZ es una zona neutral en la que un cliente podrá acceder a los servicios de la empresa sin entrar en la red interna y peligrar su integridad. Además será accesible para los miembros de la empresa de la propia red interna; pero en contrapartida desde la red perimetral no se podrá conectar a la red interna ya que podría ser un *bug* para los atacantes.

En este punto es donde situamos los servicios de la empresa que a nosotros en particular nos interesa que sean accesibles desde el exterior, lo que lo diferencia de situarlos en la propia red interna, que únicamente tendrán acceso los trabajadores.

Así pues los servicios que se ofrecen son los siguientes:

La página Web de la empresa, para que cualquier cliente o persona interesada pueda ver las posibilidades que le ofrece nuestra empresa.

El servidor de correo. En una empresa no todos los trabajadores se ubican siempre en el mismo lugar de trabajo, ya sea porque debe ir a casa del cliente, por tener que ir a otras sedes de la empresa, etc. Por eso, es importante que el trabajador pueda mantenerse al corriente de todo lo sucedido vía mail y así estar en constante comunicación. Es por eso que el servidor de correo interno se encuentra en esta misma subred, para que pueda ser consultado desde la red externa.

Por otra parte, habilitamos los servicios de conexión remota mediante la aplicación Secure Shell; pues también es útil el poderse conectar remotamente a los servidores; eso sí en este caso, únicamente a personal autorizado desde el propio *Firewall* o la red interna.

Además, en este punto instalamos el servidor de ficheros FTP para la descarga de ficheros que puedan ser de interés público.

Finalmente, otro de los servicios necesarios es el servidor de resolución de nombres DNS, el cual asocia una IP con un nombre de host; Pues es aquí donde le damos un "nombre canónico" al dominio del correo interno "@tfc.es" y a la página web de la empresa www.tfc.es. (Véase 4.8 Configuración Servidor DNS)

4.3 Configuración de la red

Una vez ilustrado el escenario que se va a diseñar y conociendo todos los dispositivos que se usan en la implementación describiremos la configuración de red que se ha llevado a cabo.

4.3.1 Configuración con Debian Sarge

Los sistema Unix disponen de un *daemon* para cargar la red y éste se configura en el `/etc./network/interfaces` que podemos ver descrito en el Anexo II-A.1. Además se adjunta la tabla de encaminamiento de cada una de las máquinas.

Para comprobar la conectividad hemos hecho un ping a los correspondientes *gateways* de cada una de las subredes. Claro que entre diferentes subredes aún no se conectan ya que no se ha configurado la herramienta que traduce las IP's.

4.3.2 Configuración en Windows 2003 Server

La configuración con Windows 2003 Server es mucho más simple que en un sistema Unix pues se configura de la misma manera que en un Windows clásico para usuarios.

Para establecer la conectividad entre las tres redes, nos dirigimos a cada una de las tarjetas de red y configuramos la IP, la máscara, la puerta de enlace y los servidores DNS. Todo ello se encuentra en el Panel de control en Conexiones de Red.

A parte, añadimos una ruta en la tabla de enrutamiento para definir la puerta de enlace a la red externa (192.168.16.16) como *gateway* predeterminado.

```
route add 0.0.0.0. mask 0.0.0.0 192.168.16.66 metric 1
```

De la misma manera que con la distribución de Debian, con los paquetes ICM comprobamos la conectividad.

4.3.3 Configuración en OpenBSD 3.6

La configuración de red en el sistema BSD es ligeramente distinta a la configuración de un sistema GNU/Linux. Para empezar debemos crear para cada tarjeta de red un archivo que incluirá la IP, la máscara y distintas opciones.
El archivo se edita en el directorio /etc con el nombre hostname.if. Éstos seguirán la siguiente estructura:

```
inet IP netmask OPTIONS
```

Para ello hemos editado los tres archivos correspondientes a cada una de las tarjetas de red que conectarán las tres subredes (véase Anexo II-A.2).

4.3.4 Configuración de la red del atacante

Como ya hemos comentado en la descripción de las redes, nuestra máquina intrusa formará parte de la red pública, por ello se le asignará una IP de forma dinámica del servidor DHCP gateway de nuestra red externa 192.168.16.66.

Paralelamente, para ver la conectividad entre subred y subred debemos añadir dos rutas en la tabla de encaminamiento para que los paquetes con destino alguna de las redes interna o DMZ tengan como *gateway* nuestro *Firewall* y no el gateway a la red externa (192.168.16.66).

```
route add 192.168.200.0 mask 255.255.255.0 192.168.16.214
metric 1
```

```
route add 192.168.100.0 mask 255.255.255.0 192.168.16.214
metric 1
```

4.4 Servidor Web

4.4.1 Servicio

La aplicación utilizada para publicar el servidor Web de la empresa es el servidor Apache.

El porqué de esta elección se debe a que este servidor es de libre distribución con un uso superior al 50% y ha servido y sigue sirviendo de referencia para muchos de los servidores comerciales que existen actualmente.

Siendo un servidor muy completo, no nos hemos dedicado a hacer un estudio profundo sobre sus capacidades, pues tampoco es el objetivo del trabajo, sino que nos hemos limitado a hacer una configuración básica.

4.4.2 Configuración

Para configurar el servicio hemos seguido los siguientes pasos:
Primero nos descargamos el paquete mediante el comando:

```
apt-get install apache
```

A continuación hemos modificado el fichero `/var/www/index.html` para escribir lo que queríamos que apareciera en la IP PÚBLICA de nuestro servidor, que se encuentra en la DMZ.

Finalmente hemos arrancado el servicio con el comando:

```
/etc/init.d/apache restart
```

Para comprobar que el proceso se está ejecutando hemos hecho un `netstat -ln` para ver que el *daemon* de Apache está escuchando por el puerto 80 (puerto del servicio Web) para cualquiera de las interfaces.

A continuación hemos accedido link de nuestra máquina-servidor Web y nos ha aparecido su contenido.

Una vez configurado el servidor DNS, la IP para acceder al servidor Web pasa a tener un nombre de host asociado (www.tfc.es).

4.5 Servidor de Correo

4.5.1 Servicio

El correo electrónico es un sistema automatizado de entrega de correo convencional, en el que intervienen tres agentes:

- El cliente de correo electrónico (remitente)-MUA (*Mail User Agent*)
- El agente de transporte del correo-MTA (*Mail Transport Agent*)
- El agente de entrega del correo (destinatario)-DA (*Destinator Agent*)

El primero es el que da formato al mensaje, lo dirige y lo entrega al agente del transporte de correo.

El Agente de Transporte o MTA acepta los mensajes de los agentes usuarios (MU) y de otros agentes de transporte. Éste encamina el mensaje por la red adecuada, resuelve los alias y el reenvío.

Finalmente, el Agente DA entrega los mensajes a un destino accesible para el receptor.

Para configurar este servicio utilizamos Sendmail como agente de transporte pues es el más utilizado y el que mejor se integra en sistemas UNIX.

Sendmail como MTA es un daemon del sistema que permite enviar y recibir correo SMTP (*Simple Mail Transfer Protocol*), protocolo de mensajes entre MTA's. Para ello, Sendmail se queda como proceso residente escuchando el puerto 25, admitiendo y realizando las conexiones SMTP cuando sea necesario.

4.5.2 Configuración

Sendmail es un programa complejo de configurar, consta de varios archivos que debemos editar (véase Anexo II-B.1):

- `/etc/mail/sendmail.cf`
- `/etc/mail/sendmail.mc`
- `/etc/mail/local-host-names`
- `/etc/mail/relay-domains`
- `/etc/mail/access`
- `/etc/aliases`

El archivo de configuración de Sendmail, `/etc/mail/sendmail.cf` es difícil de editar directamente por eso basta con modificar el archivo de configuración `/etc/mail/sendmail.mc` y procesarlo con la utilidad m4. Éste archivo ofrece un método para identificar y configurar los componentes que deseamos de manera más sencilla.

Una vez editado el archivo sendmail.mc generamos el correspondiente sendmail.cf mediante la siguiente instrucción:

```
m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
```

A continuación debemos establecer dominios a administrar en el archivo `/etc/mail/local-host-names`.

El siguiente paso es establecer los dominios permitidos para poder enviar correo, para ello editamos el archivo `/etc/mail/relay-domains`.

Finalmente hemos definido una lista de control de acceso en el archivo `/etc/mail/access` en el que incluimos las IP's locales del servidor, la lista negra de direcciones de correo, dominios e IPs denegados.

Para que los cambios tengan efecto debemos compilar `/etc/mail` con el comando: `make`

Por otra parte, en el archivo `/etc/aliases` definimos el alias para la cuenta de root a donde redireccionar el correo pues no es conveniente estar autenticando la cuenta de root a través de la red para revisar los mensajes originados por el sistema.

4.6 Servidor Secure Shell

4.6.1 Servicio

Secure Shell es una aplicación SEGURA que permite la conexión remota a una máquina a través de una red TCP/IP. Su antecesor protocolo (aunque aún muy utilizado actualmente) es Telnet (*Terminal Networking*). El origen de este protocolo vino para que cualquier persona pudiera conectarse directamente al servidor de correo desde donde fuera, claro que hoy en día, este uso no es tan común existiendo los protocolos de descarga de correo POP3 e IMAP.

La principal diferencia entre ambos protocolos es que Secure Shell es más seguro, puesto que Telnet tiene ciertas vulnerabilidades que SSH no, he aquí el porqué de nuestra elección. En Telnet se envía toda la información en texto plano por la red, de tal manera que un ***sniffer*** sería capaz de capturar el nombre de usuario y la contraseña; sin embargo ssh hace una encriptación de los datos.

La aplicación Secure Shell permite conectarnos a cualquier máquina y así poderla dirigir, siempre pasando por un sistema de autenticación de *login* y *password*. Además otra de las mejoras con respecto Telnet es que existe la posibilidad de conectarse como *root*.

4.6.2 Configuración

Primero descargamos el paquete Secure Shell.
`apt-get install ssh`

En este momento se tiene la opción de instalarse el cliente o servidor y cliente de la aplicación. Luego el cliente ssh con el comando.

Iniciamos el daemon como en todos los servicios y comprobamos haciendo un `netstat –ln` que se ejecuta escuchando por el puerto predeterminado (#22). A partir de aquí una máquina con la aplicación de cliente Secure Shell ya puede conectarse haciendo:

`ssh –l host IP máquina`

Para mayor seguridad, una vez se conecta remotamente, la máquina remota avisa que la IP de ese usuario cliente ssh quedará registrada en la lista de *hosts*, cifrada con RSA; será entonces cuando el cliente deba escribir su *password* y así poder entrar en la máquina.

4.7 Servidor FTP

4.7.1 Servicio

FTP (***F**ile **T**ransfer **P**rotocol*) es un protocolo para un servicio de transferencia de ficheros. Se utiliza en modo cliente-servidor: conectados a un ordenador remoto. Según el tipo de conexión que se establezca entre ambos el cliente puede actuar en modo pasivo o modo activo:

Modo Activo

Se establecen dos conexiones distintas. En primer lugar se abre una conexión para la transmisión de comandos (desde cualquier puerto de nuestro ordenador inferior a 1024 hacia el puerto 21 del servidor) y por esa misma conexión, se indica al servidor cual es el puerto (distinto) de nuestro ordenador que está a la escucha de los datos. Entonces, al descargarnos algún archivo es el servidor el que inicia la transmisión de datos, desde su puerto 20 al puerto que le hemos indicado.

Modo Pasivo

La aplicación FTP cliente es la que inicia el modo pasivo, de la misma forma que el modo activo. El cliente FTP indica que desea acceder a los datos en modo pasivo y el servidor proporciona la dirección IP y el puerto aleatorio, sin privilegios (mayor que 1024) en el servidor. Luego, el cliente se conecta al puerto en el servidor y descarga la información requerida.

En nuestro caso configuraremos las reglas relacionadas con el servicio FTP del *Firewall* en modo activo, ya que nuestro servidor FTP responderá únicamente a los puertos 20 y 21, puesto que el resto de puertos (<1024) estarán bloqueados.

4.7.2 Configuración

El servidor FTP que vamos a utilizar es ProFTPd pues es de los más seguros en cuanto a servidores FTP de GNU/Linux.

Si partimos de la base en que este servidor no va a ser constantemente utilizado por los clientes, puesto que se usará más la descarga con el servidor Web, lo configuramos como servidor inetd, ya que no merece la pena mantener Proftpd funcionando constantemente, siendo que provocaría un importante consumo de recursos.

El fichero `/etc/proftpd.conf` lo hemos configurado como servidor privado y anónimo; de manera que podrán acceder los usuarios de la red externa (como anónimos) y los trabajadores de la empresa con su nombre de usuario. Esto se debe a que al tener el *Firewall* detrás ya actuará de filtro, por eso no se ha configurado únicamente como servidor privado (véase Anexo II-B.2).

Para comprobar el funcionamiento del servidor, hacemos un Ftp a 192.168.200.2 (DMZ). Primero probamos entrando como usuario ya registrado desde una máquina de MS2.

Podemos ver que primero debemos autenticarnos con el *login* y *password*. A continuación nos descargamos el archivo Prueba.doc con el comando
get prueba.doc, pues los archivos se encuentran en el directorio /home/ftp.

A continuación hacemos lo mismo pero entrando como ‘anonymous’ y así ver que un usuario no registrado puede también descargarse archivos.

Como apunte, decir que hemos configurado el servidor de manera que los clientes no tengan permiso para subir ficheros y así evitar muchas amenazas de seguridad.

4.8 Servidor de Nombres DNS

4.8.1 Servicio

Todas las máquinas de una red TCP/IP se identifican por una dirección IP de 32 bits. Siendo que es mucho más fácil recordar el nombre de una máquina que recordar una ristra de números configuramos un servidor que asocie una IP con un nombre de máquina. El estándar utilizado en Internet es el DNS (*Domain Name System*).

4.8.1.1 Tipos de Servidores DNS

Un servidor DNS funciona como una red jerárquica de servidores, dedicados exclusivamente a la traducción de direcciones.

Si nos concretamos en la función de un servidor DNS podemos decir que es una base de datos con la lista de nombres e IPs de los *hosts* a los cuales da servicio. Como hemos comentado, la red de servidores DNS es jerárquica, de manera que existen varios tipos de servidores:

- Servidores esclavos: Tienen información de peticiones ya resueltas con lo que son respuestas etiquetadas como "no autorizadas".
- Servidores master (primarios): Ofrecen respuestas autorizadas porque tienen una copia maestra de los datos de la zona.
- Servidores *forwarding*: Centralizan las peticiones
- Servidores raíz: Son los servidores de primer nivel que responden cuando se busca resolver un dominio no resuelto en los servidores de niveles jerárquicos más bajos.

4.8.2 Configuración

En nuestro caso hemos configurado en la DMZ un servidor master. Para ello nos descargamos el paquete BIND (*Berkeley Internet Name Domain*), que nos proporciona un servidor de nombres llamado *named* y una librería de resolución de DNS.

El fichero de configuración principal del BIND es named.conf. Es aquí donde tenemos que declarar los ficheros de nuestras nuevas zonas (véase Anexo II-B.3).

A continuación creamos los ficheros de zona `etc/bind/tfc.es` y 192.168.200.in-addr.arpa.zone que contienen información sobre el espacio de nombre particular. Estos se caracterizan por contener unas directivas y unos registros definidos y explicados en los ficheros correspondientes.
Finalmente, una vez configurados los tres ficheros probamos el correcto funcionamiento del servidor. Para ello reiniciamos el *daemon*:

```
/etc/init.d/bind restart
```

Luego, probamos una resolución de nombres con el comando nslookup, de manera que podemos comprobar nuestro servidor DNS asociado a la IP de la máquina y ver que está escuchando por el puerto #53.

Por otro lado, como nuestro servidor DNS sólo tiene los dominios del servidor Web, del de correo y del de FTP de la empresa, añadimos en el fichero `/etc/resolv.conf` una segunda línea con la IP de nuestro servidor DNS alternativo (194.179.1.100). Primero se hará la consulta en el servidor DNS local de la DMZ, si el host buscado no consta en éste, se apuntará

directamente al puerto #53 del servidor 194.179.1.100 pues dispondrá de más dominios. Éste a su vez apuntará a otros servidores por encima con más información. Pues comprobamos que el servicio DNS es un sistema jerarquizado (véase Anexo II-B.3).

Para comprobar el funcionamiento:

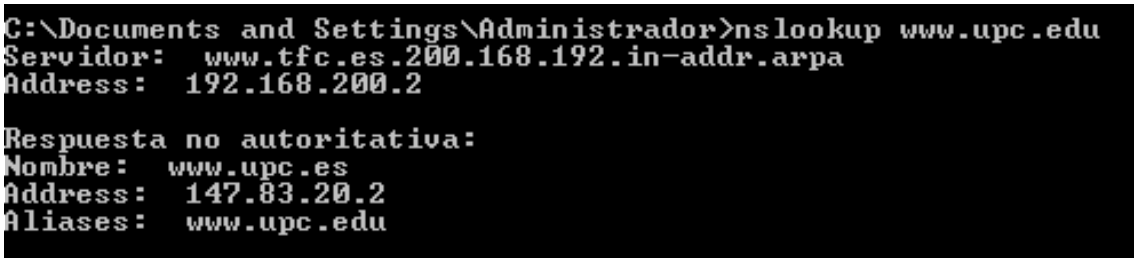


Ilustración 9. Resolución servidor DNS de la DMZ

4.9 Servidor DHCP

4.9.1 Servicio

Para que una máquina pueda trabajar en una red TCP/IP necesita como es obvio una dirección IP como mínimo, una máscara y si puede salir de su red, un *Router,* que en nuestro caso será el propio *Firewall*. La asignación manual de estos parámetros puede llegar a ser una tarea pesada y propensa a errores, sobretodo si lo tenemos que hacer para una empresa con un número importante de trabajadores. Además cualquier cambio de direcciones que se tenga que hacer, como por ejemplo pasar de IPs públicas a privadas, o las posibles manipulaciones indeseadas de algunos usuarios provocan una repetición del proceso a grande o pequeña escala. O sea que hacer la asignación de IPs a un número de nodos de manera manual es inviable.

4.9.2 Configuración

Para empezar, debemos escoger la aplicación. Podríamos trabajar con BOOTP (*Bootstrap Protocol*) o bien DHCP (*Dynamic Host Configuration Protocol*). Siendo BOOTP su antecesor y teniendo en cuenta que éste se ideó para estaciones de trabajo sin disco y por lo tanto sin capacidad para almacenar los parámetros, escogemos indudablemente el paquete d.C..
Nos lo descargamos mediante la herramienta apt-get.

```
apt-get install dhcp
```

A continuación creamos el fichero `/etc/dhcpd.conf` de configuración desde donde lee el servidor describiendo la dirección de subred (192.168.100.0), la máscara (255.255.255.0), el tiempo máximo (para que el cliente DHCP reciba la IP 7200 s) el rango de IPS a ofrecer (192.168.100.3-192.158.100.254) y el

servidor DNS asignado a esta red; el cual será el ubicado en la DMZ de la empresa (192.168.200.2). Véase Anexo II-B.4

Cabe decir, que si tuviésemos más máquinas y fuera de nuestro interés, se podría especificar para una dirección MAC a IP determinada.

Una vez escrito el fichero de configuración arrancamos el servicio desde el directorio `/etc/init.d` en el que se carga de nuevo el sistema.

```
/etc/init.d/dhcp start
```

Finalmente para comprobar el buen funcionamiento del servidor, conectamos una máquina a la subred 192.168.100.0 (*switch* Trendnet) y teniendo la aplicación de dhcp cliente, vemos que dispone de una IP, un *gateway* y un servidor DNS de esa red.

5. *Firewall*: Iptables sobre Debian Sarge

Una vez situados en todo el contexto del Proyecto empezamos a estudiar el primer *Firewall* software basado en *Iptables* con la distribución Debian Sarge del sistema operativo GNU/Linux. En este capítulo desarrollaremos todo el estudio teórico y práctico que se ha hecho para la configuración y el análisis del *Firewall.*

En primer lugar describimos las herramientas de seguridad que nos ofrece el núcleo del sistema operativo GNU/Linux (Kernel). A continuación estudiamos la herramienta *Iptables* a nivel teórico para luego poder configurar las reglas que definen nuestro *Firewall* con las pertinentes pruebas que verifican el correcto funcionamiento. Llegados a este punto hacemos un estudio de vulnerabilidades de los servicios que se sitúan detrás del *Firewall* para ver cómo éste actúa y así luego poder hacer un análisis de los resultados.

El capítulo consta de varios apartados:

- Seguridad en el *Kernel*
- Descripción del *Firewall* con Iptables
- Configuración de las reglas del *Firewall*
- Pruebas de vulnerabilidad
- Valoración

.

5.1 Seguridad en el Kernel

La distribución de Linux que usamos es la estable de Debian Sarge ya que nos garantiza mayor seguridad.

Antes de sumergirnos en la herramienta de las *Iptables* podemos establecer ciertos parámetros del Kernel que nos ayudan a incrementar la seguridad ante los ataques explicados en el Capítulo 3.

Accediendo a `/proc/sys/net/ipv4` nos detenemos en los siguientes parámetros:

5.1.1 Paquetes ICMP

Como hemos descrito en el Capítulo 3. donde definimos los posibles ataques, una de las violaciones de servicio más comunes es la de los barridos ICMP.

Además, los paquetes ICMP echo pueden ser una herramienta de gran utilidad para cualquier atacante y con este comando se evita una respuesta a estos pings de manera que la máquina aparece como invisible.

- `icmp_echo_ignore_all=1`

De la misma manera para los paquetes broadcast.

- `icmp_echo_ignore_broadcasts=1`

5.1.2 Información de Enrutamiento

En IP *source routing*, cada paquete contiene la información del camino a seguir hasta llegar al *host* de destino. El peligro reside en que de acuerdo con el RFC 1122, el camino de ida debe ser el mismo que el de vuelta y si un intruso envía un paquete *source routing* podrá saber por qué hosts de nuestra red interna pasará; por ello deshabilitamos el parámetro correspondiente.

- `conf.eth#.accept_source_route=0`

5.1.3 Deshabilitación *Spoofing*

Haciendo referencia a los ataques de *Spoofing* del Capítulo 3, Kernel nos ofrece la posibilidad de deshabilitarlo con el parámetro de configuración *Reverse Path Filtering:*

- `conf.eth#.rp_filter=1`

De esta manera verificamos en la tabla de encaminamiento que para cada interfaz no se estén recibiendo paquetes de direcciones origen no legítimas.

5.1.4 SYN cookies

El ataque DoS por excelencia es el ataque *SYN Flood* y puede ser evitado utilizando las *SYN cookies* que es un método que consiste en continuar aceptando peticiones cuando la cola *SYN* está llena. Esto se consigue generando unos pequeños paquetes de información, las *cookies*, que contienen información sobre las cabeceras TCP de los paquetes recibidos, denegando así los paquetes originales.

Aún así, el activar las *SYN cookies* no puede proporcionar inmunidad contra ataques DoS, especialmente cuando el ancho de banda atacante es más importante del que puede gestionar el servidor.

Por otra parte es una opción muy recomendada el activar éste parámetro especialmente en la implementación de la pila TCP/IP de Linux, enfocada en el rendimiento y no en la seguridad.

El parámetro a modificar es:

- net.ipv4.tcp_syncookies=1

5.1.5 *Martians*

Habilitamos que se guarde en el registro log los paquetes que hayan sido "esnifados" de source_route (Véase apartado 5.2).

- `net.ipv4.conf.eth#.log_martians= 1`

Donde # es el número de interfaz. En nuestro caso lo habilitamos para todas.

5.2 Descripción del *Firewall* con Iptables

La aplicación de filtrado más utilizada con Linux es *Netfilter* cuya herramienta de trabajo es Iptables. *E*s un sistema de *Firewall* vinculado al Kernel de Linux, integrado en él. Funciona en base a unas reglas que permitirán bloquear o modificar la salida y entrada de todos los paquetes. Si hacemos referencia a la torre OSI este *Firewall* realiza el filtrado a nivel de red (IPv4 o IPv6) y de transporte (TCP, UDP, ICMP).

Para ponerlo en funcionamiento se ejecuta el comando Iptables en el que creamos las reglas; por eso, podemos definir un *Firewall* de *Iptables* como un simple script de shell en el que se van ejecutado las reglas.

La manera de administrar el conjunto de condiciones por las que queremos que se rija nuestro *Firewall* es mediante las cadenas, donde éstas a su vez de agrupan en tablas.

5.2.1 Cadenas

Cuando hablamos de cadenas, nos referimos a una lista de reglas agrupadas lógicamente. Cada regla de una cadena es una prueba que aplicar contra la cabecera de una IP para buscar su correspondencia.

Las cadenas contienen reglas que están numeradas a partir de uno. Se puede referir o bien por la especificación de regla o bien por el número de regla.

Una especificación de regla es el conjunto de condiciones que tiene que tener un paquete; pues la misma regla básica puede existir en múltiples cadenas (INPUT/OUTPUT y FORWARD) como explicaremos en el siguiente punto; por eso el parámetro cadena es obligatorio.

5.2.2 Tablas

Netfilter se define por tres tipos de tablas:

- Tabla *filter*: Actúa como filtro de paquetes

- Tabla *mangle*: Altera los paquetes
- Tabla NAT: Traduce direcciones de red

La tabla *filter* es la que se utiliza por defecto. Dispone de tres cadenas distintas para diferenciar el tráfico de entrada, el de salida y el que se enruta por el propio *Firewall*:

- INPUT: Se definen las reglas de los paquetes recibidos para nuestro sistema
- OUTPUT: Son las reglas de los paquetes que se generan en nuestro servidor local y son enviados.
- FORWARD Definida para los paquetes que vayan a ser enrutados.

Dado que los paquetes irán a una de las cadenas INPUT o FORWARD (pero no a ambas), cualquier regla que quiera aplicar igualmente a ambas tendrá que residir en las dos cadenas.

La tabla *mangle*, no muy utilizada se usa para alterar o modificar los bits de tipo de servicio (TOS) del encabezado del paquete IP. De la misma manera, consta de dos cadenas:

- PREROUTING: Para alterar los paquetes entrantes antes de enrutar.
- POSTROUTING: Para alterar los paquetes generados localmente antes de enrutar.

Esta tabla permite marcar los paquetes para poder ofrecer QoS con un módulo adicional pero no para priorizar dentro de la propia configuración de *Iptables*. Así pues esta tabla consta de tres parámetros de marca:

  - **TOS (*Type of Service*)**: Se utiliza para identificar el tipo de servicio del paquete. Es útil en el momento que se hace un control de policía en cuanto cómo se debe enrutar el paquete, así pues se necesita el paquete iproute2[1] para hacer estas funciones de enrutamiento.
  - **TTL (*Time to live*)**: Parámetro que se utiliza para cambiar el valor del TTL de determinados paquetes que no interesan que tengan un tiempo de vida largo.
  - **MARK**: Esta marca se utiliza cuando se quieren hacer limitaciones de ancho de banda para determinado tráfico con el algoritmo CBQ[2] (*Class Based Queueing*). Del mismo modo que en los dos parámetros anteriores, se utilizará únicamente será considerada por iproute2.

51

[1] Módulo que permite el manejo de múltiples tablas de enrutado y soporta controles de policía de enrutado de acuerdo con las marcas en los paquetes.

[2] Algoritmo de formación de colas que divide el ancho de banda de una conexión de red entre varias colas o clases. A cada cola se le asigna un tráfico basándose en la dirección de origen o de destino, el número de puerto, protocolo, etcétera.

En cuanto a la tabla NAT o MASQUERADING permite añadir una conversión de IP privada a pública o al revés donde se usarán las dos cadenas adicionales, POSTROUTING y PREROUTING, que vienen definidas por el mismo criterio explicado en las cadenas de la tabla *mangle*.

Éstas se usan con DNAT y SNAT (destino y fuente NAT). Para entenderlo mejor, se puede pensar en que estas dos cadenas se encuentran fuera de la máquina, de modo que podemos imaginar que la conversión se hace justo antes de que el paquete llegue a la interfaz de entrada o salida.
En modo gráfico se ve de la siguiente manera:

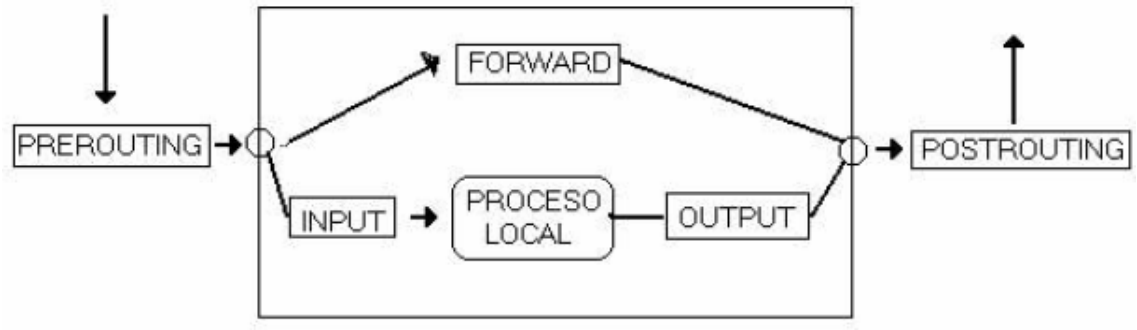


Ilustración 10. Flujo de paquetes a través de Iptables

Todos los paquetes fluyen de izquierda a derecha y pasan a través de la cadena INPUT o la cadena FORWARD, pero nunca por ambas a la vez.

Cuando un paquete llega a un interfaz, primero llega a la cadena PREROUTING (si existe). Aquí las direcciones de destino se cambian (DNAT) para indicar otro servidor diferente. Si por ejemplo, el tráfico al puerto 80 del servidor local se va a mandar a un servidor interno, el destino se cambia aquí. El paquete llega entonces a la interfaz y el Kernel hace la búsqueda en la tabla de encaminamiento para ver a dónde tiene que ir el paquete (al servidor local o a otra interfaz).

Entonces el código de *Netfilter* toma el paquete basándose en su destino (servidor local o interfaz de salida) y ejecuta la cadena INPUT o la FORWARD. Si el paquete pasa por la cadena FORWARD y no se rechaza, se le vuelve a pasar por la tabla de encaminamiento del Kernel y se le envía a la interfaz apropiada.

Mientras el paquete sale de la interfaz, *Netfilter* hace pasar el paquete otra vez por las reglas de POSTROUTING.

Todos los paquetes que vienen o van experimentar el mismo proceso básico. El proceso también se aplica a paquetes que utilicen la interfaz de servidor local, que pasan por las reglas INPUT y OUTPUT.

5.2.3 Sintaxis de las reglas

La sintaxis básica a la que responden todas las reglas para definir los paquetes es la siguiente:

```
iptables [-t tabla] cadena regla [opciones]
```

Donde:

- [–t tabla] es el tipo de tabla si es filter, NAT o mangle.
- La cadena (INPUT, OUTPUT, FORWARD...)
- A continuación se escribe la regla que se quiere definir, pues viene en función de lo que el administrador desee configurar. Se basa en unos parámetros como son el protocolo del paquete (`-p`), el puerto destino (`-dport`) u origen (`-sport`), etc.

- Finalmente, como [`opciones`] aparecen una serie de parámetros que se pueden definir como por ejemplo añadir una regla (--append), borrar una cadena (--delete), definir la política de la cadena (--policy), listar las regla de una cadena (--list),etc. Entre todas ellas, hay una opción llamada (--jump) con la que podemos especificar la acción que se va a llevar a cabo con esa regla.

Mediante el comando de ayuda: `man iptables` se puede visualizar con mayor detalle todas las posibilidades que nos ofrece esta herramienta.

5.2.4 Acciones para cada regla

Como hemos podido ver en el anterior apartado 5.2.3, cada una de éstas describen las características del paquete diciendo el tipo de protocolo, la interfaz de salida o de entrada, la IP origen o destino, etc; es decir existen una serie de parámetros con los que definir la regla, `[opciones];` pero a parte debemos especificar el objetivo de esa regla.

Hay varias acciones que se pueden ver explicadas en el manual de *iptables*, pero queremos hacer destacar cuatro de ellas debido a su uso:

- ACCEPT: El paquete se acepta y sale de la cadena INPUT
- DROP: El paquete se descarta
- LOG: Permite que los paquetes con los que se ha establecido una correspondencia sean registrados. (Útil para registrar ataques maliciosos).
- REJECT: Se envía un paquete ICMP de rechazo.

La principal diferencia entre el modo DROP y REJECT es que cuando se rechaza un paquete, éste se descarta y se devuelve un mensaje de error ICMP al remitente. En cambio, cuando se deniega un paquete, simplemente se descarta el paquete sin ningún tipo de notificación al remitente.

La denegación es casi siempre la mejor elección, debido a tres razones:

Primero, enviar una respuesta de error duplica el tráfico de red. La mayoría de los paquetes se descartan porque son malévolos, no porque representen un intento inocente de acceder a un servicio que no se le ha ocurrido ofrecer.

Segundo, cualquier paquete al que responda se puede usar en un ataque por denegación de servicio. Como tercera y última razón, cualquier respuesta, incluso un mensaje de error ofrece información potencialmente útil a quien podría ser un *hacker*.

Por todo ello, adoptamos la política regida por DROP y no por REJECT.

5.2.5 El entorno Iptables

Antes de configurar el *Firewall* debemos familiarizarnos en los parámetros de configuración de las reglas en el entorno de *Iptables*, siendo que es una aplicación muy completa requiere cierta habilidad para configurar las reglas; por eso, corroboraremos en la práctica todo lo comentado en los puntos anteriores.

Para empezar, mediante el comando: `iptables –L` podemos ver que por defecto la política de seguridad con la que se rige la herramienta es de aceptar todos los paquetes.

- Iptables –P INPUT ACCEPT
- Iptables –P OUTPUT ACCEPT
- Iptables –P FORWARD ACCEPT

Si no se especifica ningún parámetro se da a entender que se aceptará todo paquete de la interfaz que provenga, sea cual sea su protocolo o su puerto origen y destino, etc. Por ello, lo modificamos en modo DROP dado que nuestra política a seguir es la de denegar todo.

Por otro lado, si listamos las cadenas de la tabla nat mediante el comando: `iptables –t nat –L` vemos que también las tres cadenas están en modo ACCEPT y no hay ninguna regla definida. En este caso, ninguna de las tres subredes (interna, externa o DMZ) se podrá intercomunicar puesto que no se aplica la traducción de IP's. Para ello debemos hacer un enmascaramiento de estas a través de la "acción" MASQUERADE.

```
iptables –t nat –A POSTROUTING –s 192.168.100.0/24 –j MASQUERADE #LAN interna
iptables –t nat –A POSTROUTING –s 192.168.200.0/24 –j MASQUERADE #DMZ
```

Ahora sí que existe la conectividad entre ellas, pues con estas reglas decimos que para cualquier máquina de una de las dos subredes aplique una traducción de IP en la salida (POSTROUTING).

Por otra parte, hemos querido comprobar a nivel práctico que las cadenas INPUT y OUTPUT son independientes de la cadena FORWARD (como hemos explicado teóricamente en el punto (5.2.2. Tablas).

Si partimos de la base en poner las cadenas INPUT y OUTPUT en modo DROP y la cadena FORWARD en modo ACCEPT, podemos seguir conectándonos desde la red interna a la red externa, independientemente de si la cadena OUTPUT o INPUT está habilitada; es por eso que no será necesario repetir las reglas para ambas cadenas.

Luego probamos el caso contrario y efectivamente vemos que desde la red interna o desde la DMZ no podemos acceder a la red externa, si no es habilitando la cadena FORWARD.

A continuación nos disponemos ya a configurar las reglas de nuestro *Firewall*

5.3 Configuración de las reglas del *Firewall*

Para empezar, nos disponemos a crear un archivo llamado *Firewall* en el que configuraremos las reglas. Haremos que se cargue cada vez que se reinicia la máquina desde el directorio `/etc/init.d`.

A parte, debemos darle los permisos de lectura y escritura necesarios.
`chmod 0755 /etc/init.d/firewall`

Por otra parte, para administrar el runlevel utilizamos el programa de Debian update-rc.d.

Primero borramos todos los enlaces en los directorios `rc.d` al guión `/etc/init.d/firewall`.
`update-rc.d –f /etc/init.d/firewall remove`

A continuación creamos los enlaces que inician *Firewall* en los runlevels 2345.
`update-rc.d /etc/init.d/firewall start 99 2 3 4 5 .`

Una vez creado el fichero configuramos el *Firewall* en base a la política de seguridad definida (véase Anexo II-C.1).

5.4 Pruebas de vulnerabilidad

Para comprobar el correcto funcionamiento del *Firewall* accedemos a los servicios desde cada una de las subredes para verificar que el *Firewall* habilita o deshabilita el tráfico en base a las reglas definidas en su configuración; o sea que intentamos violar aquellos accesos no permitidos y comprobamos que habilita el tráfico permitido.

Para ello hemos hecho las mismas pruebas que hacíamos al comprobar que los servicios funcionaban correctamente pero ahora desde las 3 subredes y así ver qué tráfico deja o no pasar nuestro *Firewall*.

Una vez comprobado el correcto funcionamiento del *Firewall* comprobando si sólo permite el paso del tráfico estipulado por las reglas nos disponemos a hacer un escaneo de vulnerabilidades con Nessus en los servicios de la DMZ.

Tabla de los servicios del host (DMZ)

Los servicios que están habilitados por el *Firewall Iptables* son los que detectan Nessus y los lista junto con el tipo de mensajes que tiene sobre cada uno.

Análisis del *Host*	
Puerto/Servicio	**Mensaje según el puerto**
ssh (22/tcp)	Notas de seguridad
smtp (25/tcp)	Agujero de seguridad
domain (53/udp)	Notas de seguridad
www (80/tcp)	Avisos de seguridad
pop3 (110/tcp)	Notas de seguridad
ftp (21/tcp)	Avisos de seguridad
general/tcp	Avisos de seguridad
general/udp	Notas de seguridad

Tabla 2. Análisis del escaneo de puertos de la DMZ

Gráfica del tráfico en la red

En esta gráfica adjunta podemos ver el tráfico que generan los servicios y sus respectivos puertos (eje X) en base al número de ocurrencias (eje Y). Podemos comprobar que hay una sola ocurrencia por servicio puesto que se analiza un sólo *host* que es el que dispone de ese servicio.

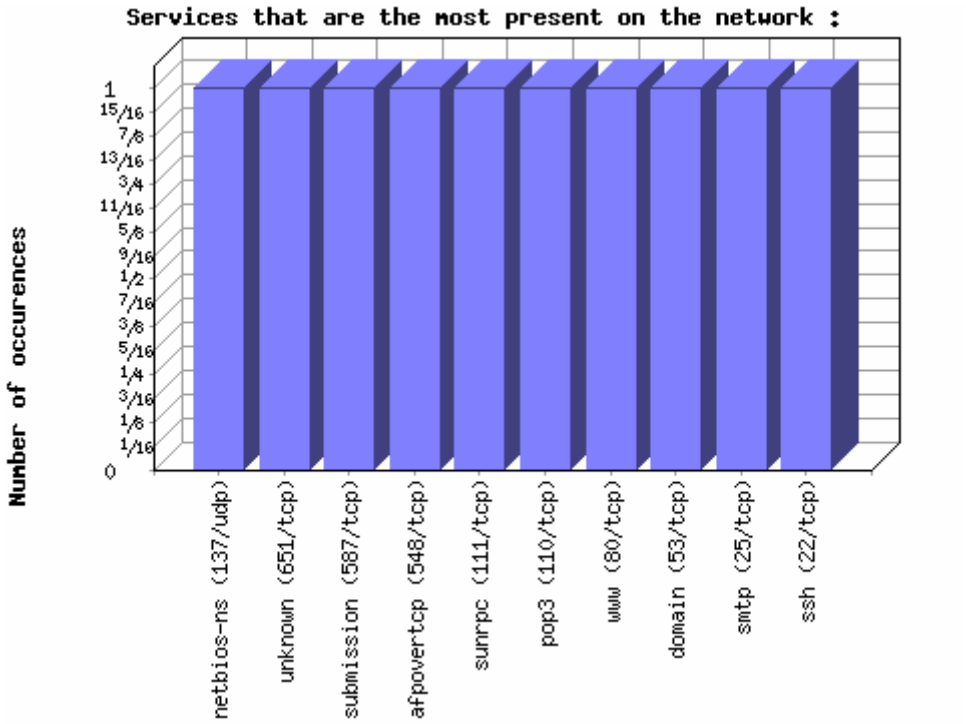


Ilustración 11. Análisis del tráfico de la red perimetral con Iptables

Podemos comprobar que además de los servicios que hemos configurado, Nessus ha detectado tráfico que está habilitado por defecto:

- Netbios-ns(#137): Tráfico del servicio NetBIOS (*Network Basic Input/Output System*) que permite que las máquinas de una LAN compartan recursos teniendo asignado un nombre NetBIOS.

- Tráfico de submission (#587): Es tráfico que proviene del servidor de correo, pues el cliente local en lugar de invocar directamente a sendmail para enviar correo electrónico desde el servidor, abre el puerto 587 para enviar un mensaje. Lo podemos deshabilitar mediante la opción “no_default_msa” en el archivo sendmail.mc.

- Sunrpc (#111): Es el *portmapper* de *sunrpc* y se instala de manera predeterminada como parte del sistema base de Debian porque no se sabe cuando un programa de usuario necesitará usar ***RPC*** (*Remote Procedure Call*) para funcionar correctamente. Siendo que se han encontrado agujeros de seguridad en este tipo de servicios, lo deshabilitaremos eliminando el paquete *portmap* de Debian.

Gráfica de los riesgos de seguridad

Como podemos ver en la Ilustración 12 el porcentaje de alto riesgo (agujero de seguridad) es del 5% y el de bajo riesgo es del 75%. Este 5% es el que debemos corregir para que nuestro *Firewall* sea efectivo por ello nos disponemos a resolver estas debilidades.

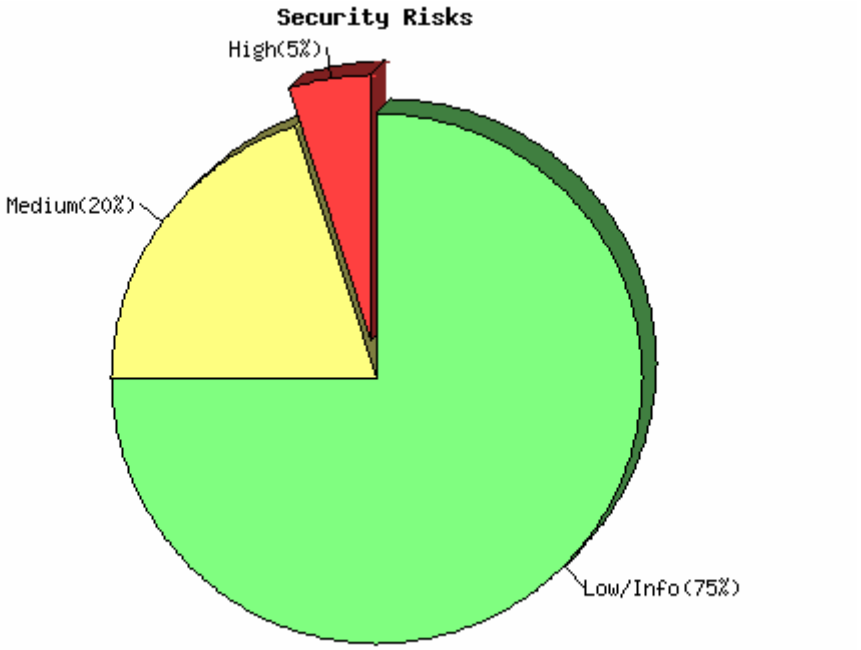


Ilustración 12. Gráfica del porcentaje de riesgo en los servicios del host con *Iptables*

Agujeros de seguridad en el escenario

Como vemos en la Ilustración 13, según el eje de las X (nombre del servicio) y el eje de las Y (número de agujeros de seguridad), el único agujero que Nessus ha detectado se ve provocado por el servicio de correo.

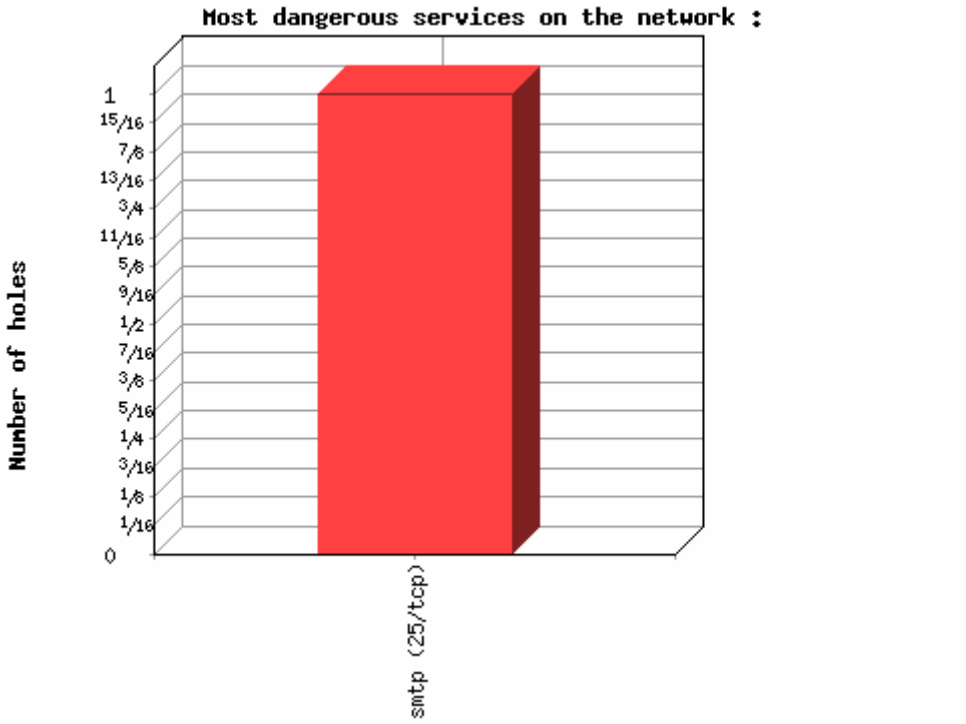


Ilustración 13. Gráfica de los servicios con agujeros de seguridad

Vulnerabilidades en los servicios

En base a la tabla 2 adjunta en el Anexo III A podemos sacar los siguientes informes de vulnerabilidad para cada uno de los servicios:

Servidor Web

Mensaje	Contenido	Problema	Solución
Notas	Habilitados métodos TRACE/TRACK (rastreo y localización)	Permite que el atacante envíe un paquete de rastreo y si el servidor lo tiene habilitado le responde y permitirá que éste lea la *cookie*	Añadimos en el fichero de configuración del Apache /etc/apache/httpd.conf: RewriteEngine on RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK) RewriteRule .* - [F]
	Apache/1.3.33 (Debian GNU/Linux)	Con un simple telnet al puerto 80 el servidor responde dando su versión	Cambiamos la opción del servidor Apache 'ServerTokens Prod' ServerTokens min' Así dejamos de publicar información sobre el servidor utilizado.
	Servidor web en puerto 80	-	-
	Servidor proxy en puerto 80	-	-

Aviso	/cgi-bin, /icons, /images	Debemos asegurarnos que pertenecen a un estándar de seguridad	Estos archivos que se encuentran en el directorio /usr/lib/ y /usr/share y que muestran que el servicio está abalado un estándar de seguridad, Mailman (software GNU).
Bug	-	-	-

Tabla 3. Vulnerabilidades en el servidor Web con *Iptables*

Si aplicamos cada una de las soluciones propuestas por Nessus y cargamos el *daemon* de nuevo para que los cambios tengan efecto, al hacer un telnet al puerto 80 y luego un GET comprobamos que ya no aparecen los datos de la versión del servidor y protocolo.

Servidor de Correo SMTP

Es en este servicio donde Nessus detecta un agujero de seguridad en el que debemos detenernos para corregirlo y evitar que sea una amenaza a la violación de seguridad.

Mensaje	**Contenido**	**Problema**	**Solución**
Notas	Servidor de correo SMTP en puerto 25	-	-
Aviso	El servidor SMTP responde a los comandos EXPN y VRF.	El comando EXPN puede utilizarse para encontrar los alias de direcciones de correo incluso el nombre completo del destinatario. El comando VRF se puede utilizar para comprobar si es válida una cuenta. El servidor no debe permitir que usen estos comandos usuarios remotos.	Añadir la opción en el fichero /etc/sendmail.cf O PrivacyOptions=goaway
Bug	No reacciona ante el comando: MAIL FROM |testing	Permite que cualquiera pueda usar comandos en este host	Actualización de sendmail

Tabla 4. Vulnerabilidades en el servidor de correo con Iptables

Servidor de Secure Shell

Mensaje	Contenido	Problema	Solución
Notas	Servidor de SSH en el puerto 22	-	-
	Versión SSH Remoto: SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4 Mecanismos de autenticación soportados: publickey,keyboard-interactive	-	-
	La versión del *daemon* SSH soporta las siguientes versions de protocolo SSH: . 1.99 . 2.0	-	-
Aviso	-	-	-
Bug	-	-	-

Tabla 5. Vulnerabilidades en el servidor SSH con Iptables

Servidor FTP

Mensaje	Contenido	Problema	Solución
Notas	Versión: ProFTPD 1.2.10 Server	-	-
	Servidor de FTP en el puerto 21	-	-
Aviso	Versión obsoleta a la 1.2.10.	Es posible ver los usuarios que se pueden conectar al servidor remoto.	Actualizar la versión
		Las directivas ftpshut y SQLShowInfo son un arma para el atacante.	
Bug	-	-	-

Tabla 6. Vulnerabilidades en el servidor FTP con Iptables

Servidor DNS

Mensaje	Contenido	Problema	Solución
Notas	Servidor DNS en el puerto 53	-	-
	Servidor BIND. Permite el acceso remoto de usuarios para conocer la versión del *daemon* que se ejecuta, pues se almacena en 'versión.bind' La versión de BIND remoto es: 8.4.6-REL-NOESW	El atacante tenga información adicional que le pueda servir de ayuda.	Usar la directiva 'version' en el parámetro 'options' y así bloquear 'version.bind', aunque no se bloquean las peticiones.
Aviso			
Bug	-	-	-

Tabla 7. Vulnerabilidades en el servidor DNS con Iptables

Del mismo que para el resto de servicios corregimos la vulnerabilidad aplicando la solución propuesta por Nessus y comprobamos que en el siguiente escaneo deja de presentar la notificación o aviso pertinente.

Una vez descritas las vulnerabilidades de los servidores y habiendo corregido aquellas que pueden ser una amenaza para la seguridad del sistema podemos sacar conclusiones sobre el comportamiento de nuestro *Firewall*.

5.5 Valoración

Sistema Operativo

La protección de una red mediante el *Firewall Iptables* requiere considerar el comportamiento que debe llevar a cabo cada uno de los paquetes IP que entran o salen. Para su configuración es necesario tener cierta habilidad con el sistema operativo de GNU/Linux ya que difiere del sistema operativo comúnmente usado por los usuarios.

De todas maneras es un Sistema Operativo cada vez más utilizado hoy en día, como alternativa a Windows ya que en algunos casos es mucho más apto. Por

otra parte, los conocimientos requeridos de GNU/Linux para trabajar con *Iptables* no son muy extensos.

Política de seguridad

La política de seguridad que sigue *Iptables* es de aceptación de todo tráfico entrante y saliente. Siendo que esta política no es la más segura, la hemos modificado por la que deniega todo tráfico entrante y/o saliente.

Configuración

La configuración de las reglas la podemos calificar como una configuración sencilla, pues por lo general todas las reglas siguen el mismo modelo, cambiando los parámetros TCP/IP. La dificultad viene cuando hay un gran número de reglas por configurar debido a que el mantenimiento del *Firewall* resulta ser mucho más complejo y pesado. Siendo que esto es fácil que ocurra ya que debemos configurar para cada tipo de tráfico una regla que le haga referencia; es recomendable seguir una política de negación de tráfico para sólo habilitar única y exclusivamente el tráfico deseado y de esta manera minimizar el máximo número de reglas posibles. Además de que sin lugar a dudas esta política es la más segura.

Por otro lado, es fundamental tener claro cómo funciona el establecimiento y la finalización de las conexiones en una red. Por ejemplo con el servidor POP3; dejamos pasar el tráfico por el puerto 110 pero debemos tener en cuenta que únicamente permitimos el tráfico del cliente al servidor, no en el sentido inverso.

Tipo de filtrado

Como ya hemos comentado a lo largo de este capítulo *Iptables* inspecciona los paquetes a nivel TCP/IP. Por eso nuestras reglas filtran en base a las direcciones origen o destino a los puertos origen y destino o al tiempo de ida y vuelta del paquete o TTL (***T**ime **T**o **L**ive*) y no en base a la capa de aplicación, lo que impide hacer un filtrado menos granulado.

Características

Una de las ventajas que caracteriza esta herramienta es que permite la instalación de *stateful Firewalls.* Esto significa que el *Firewall* tiene una memoria para cada una de las conexiones, esencial para la configuración de FTP activo (Véase apartado 4.7.1) y para el servicio DNS. Además, de que podemos bloquear algunos de los ataques DoS más comunes sin tener que incluirlo como regla en el *Firewall*. (Véase apartado 5.1).

En cuanto a la carga de las reglas, ésta es muy flexible y fácil de configurar, lo que mejora las antiguas *ipchains*, a parte de que el tipo de *Firewall* de *Packet Filter* permite que sea mucho más rápida debido a que el filtrado es menos exhaustivo.

Como ya hemos dicho anteriormente, este tipo de *Firewalls* tiene la carencia de no inspeccionar el *payload* del paquete, algo que sí hacen los *Proxies* (Véase Apartado 2.2.3.1 está experimentando un nuevo módulo de Kernel llamado *string matching match* que permite una inspección limitada del *payload* del paquete impidiendo la entrada de ciertas peticiones al servidor Web que no interesan. Este módulo amplía ligeramente las fronteras de un filtrado de este tipo pero no llega a la cima para inspeccionar los paquetes más allá de este nivel. Cabe decir que este estudio no lo hemos hecho en la práctica porque tampoco nos ha sido imprescindible para el estudio de nuestro *Firewall*.

Calidad de Servicio (QoS)

Si hacemos referencia a la Calidad de Servicio, la herramienta *Iptables* en sí no ofrece la posibilidad de priorizar el tráfico, únicamente permite marcar el tráfico con la tabla mangle. Por defecto los paquetes que entran o salen a través del *Firewall* siguen una disciplina de cola FIFO y para poder priorizar el tráfico es necesario el paquete Iproute2 que añade control de tráfico TCP/IP.

Análisis de vulnerabilidades

Por lo que respecta al análisis de vulnerabilidades con Nessus, comprobamos que no cubre al 100% la seguridad exigida puesto que se ha encontrado un agujero de seguridad en el servidor de correo.

A modo de conclusión

En definitiva, el *Firewall* configurado con *Iptables* es poco granulado ya que los parámetros de filtrado no van más allá de la cabecera IP con lo que *Iptables* es un buen comienzo para crear un *Firewall* más o menos sencillo en una red.

6. *Firewall*: ISA Server 2004 Enterprise sobre Windows 2003 Server

Del mismo modo que en el capítulo anterior desarrollaremos todo el estudio teórico y práctico que se ha hecho para la configuración y el análisis del *Firewall* de Microsoft.
En un primer apartado hacemos una descripción de las características del propio *Firewall*. A continuación, dedicaremos un segundo apartado lo para definir la configuración llevada a cabo y finalmente en un tercer punto hacemos el estudio de vulnerabilidades.

6.1 Descripción del *Firewall* ISA Server

*Microsoft **I**nternet **S**ecurity and **A**cceleration Server* (ISA) es un producto que tiene como funcionalidades las de servidor *Firewall*, servidor caché o ambas integradas. Este *Firewall* permite filtrar los paquetes a nivel de aplicación de la torre OSI lo que permite hacer un filtrado mucho más granulado.

Este tipo de filtrado permite utilizar los puertos "universales" para pasar todo el tipo de tráfico. Es decir, si en un cortafuegos de una empresa no dejan abierto un puerto determinado, con ISA Server es posible pasar el tráfico del protocolo correspondiente por uno de los puertos que sí tenga habilitado.

El estudio de nuestro *Firewall* se basará en las funcionalidades como servidor *Firewall*; es decir, cuando ISA es capaz de asegurar la red mediante la configuración de reglas que controlen las comunicaciones entre varias redes.

Algunas de las características más relevantes son:

- Filtrado a nivel de aplicación: ISA Server 2004 va más allá del filtrado básico al controlar el tráfico específico de una aplicación con filtros de datos, comandos y aplicaciones. Mediante el filtrado inteligente de VPN, HTTP, FTP, SMTP, POP3, DNS, conferencia H.323, transmisión de multimedia y tráfico RPC, ISA Server 2004 puede aceptar, rechazar, redirigir y modificar el tráfico en función de su contenido.

- Filtración dinámica de paquetes: Abre puertos en solicitud a la respuesta del usuario y cierra los puertos cuando termina la comunicación.

- Publicación de servidores: Permiten el acceso a servidores de una red interna o perimetral desde una red externa agregando un nivel de seguridad. Las reglas de publicación en servidores protegen los servidores internos del acceso no deseado por parte de usuarios externos. El filtrado inteligente de aplicaciones protege a todos los servidores publicados de ataques procedentes del exterior.

- Redes privadas virtuales integradas: ISA Server 2004 permite proporcionar acceso remoto seguro basado en estándares para conectar las sucursales y los usuarios remotos a las redes corporativas. Además da la posibilidad de ofrecer prioridades en el ancho de banda de manera que ofrece calidad de servicio (QoS).

- Detección de Intrusos. Integra una herramienta de detección que puede alertar sobre ataques a los que pueda estar sometido la red interna, por ejemplo si alguien intenta hacer una exploración de puertos.

- Autenticación. Se pueden establecer diferentes modos de autenticación, ya sea utilizando la autenticación de Windows o a través del cifrado ***hashing***. Además ISA Server puede autenticar a los usuarios con ***RADIUS*** (***R****emote* ***A****uthentication* ***D****ial-****I****n* ***U****ser* ***S****ervice*).

- Servidor de almacenamiento en caché Web: ISA Server 2004 usa un almacenamiento en caché RAM que resulta muy rápido y una caché de disco optimizada con el objeto de aumentar el rendimiento en Web, tanto en el caso de los clientes internos que tienen acceso a servidores Web en Internet como para los usuarios externos de Internet que visitan el servidor Web corporativo.

- Política de acceso. El servidor ISA se puede configurar para ejecutar reglas de acceso a protocolos, contenidos y sitios por parte de los clientes de la red interna a Internet, permitiendo o no el acceso; pues la reglas especifican el sitio y el contenido al que se puede tener acceso.

- Plataforma extensible. Escalabilidad

    - **Rendimiento por escalamiento vertical**. ISA Server está diseñado para trabajar con múltiples procesadores al optimizarse para Windows 2000. A diferencia de muchos otros productos, ISA emplea el potencial de procesamiento adicional para mejorar el rendimiento. (ISA Server Standard Edition tiene capacidad para un máximo de cuatro procesadores).

    - **Escalabilidad horizontal (*clustering* y balanceo de carga).** ISA Server utiliza Windows NLB (*Network Low Balancing*) Services de Microsoft con el *clustering* (agrupamiento de varios equipos que reciben un mismo trato) y con Enterprise Edition, puede aprovechar las ventajas del ***CARP*** (*Cache Array Routing Protocol*) para proporcionar tolerancia en fallos, alta disponibilidad, mayor eficiencia y rendimiento por medio del *clustering* de múltiples equipos de ISA Server.

6.1.1 Arquitectura servidor ISA

El servidor ISA se puede configurar como servidor independiente o como un servidor integrado en un *array*. Estos *arrays* incluyen uno o más servidores ISA que tienen la misma configuración, una administración centralizada y una política de seguridad centralizada. En nuestro caso configuramos ISA como servidor *array* y configuraremos un único servidor para analizar más a fondo todas sus posibilidades y poder permitir en un futuro el configurar más de un servidor si fuera necesario; pues de la otra manera no sería posible.

6.1.1.1 Servidores en Array

Un *array* es un grupo de servidores ISA que se utilizan para implementar tolerancia a fallos, balanceo de carga y caché distribuido. Esto permite que el *array* sea tratado y administrado como una sola entidad lógica. Si se diera el caso de instalar más de un servidor ISA, los servidores en el *array* tienen una misma configuración, permitiendo administrar centralizadamente todos los servidores en el *array* y todos los *arrays* dentro de una misma empresa. Cuando se modifica la configuración de un *array*, todos los servidores ISA que lo componen son modificados, incluyendo las políticas de acceso y las políticas de caché. Esto nos puede servir de ayuda para si en un futuro, necesitamos añadir otro *Firewall*.

Las políticas de acceso de un *array* o políticas de *array* tal cual se denomina en el programa, pueden crear reglas de acceso a sitios, contenidos, protocolos, filtrado de paquetes, publicaciones Web, etc. Estas políticas tienen como ámbito de aplicación a los sistemas que pertenecen a un mismo *array*. Por otra parte, las políticas de empresa son las que engloban las políticas de acceso de diferentes *arrays*.

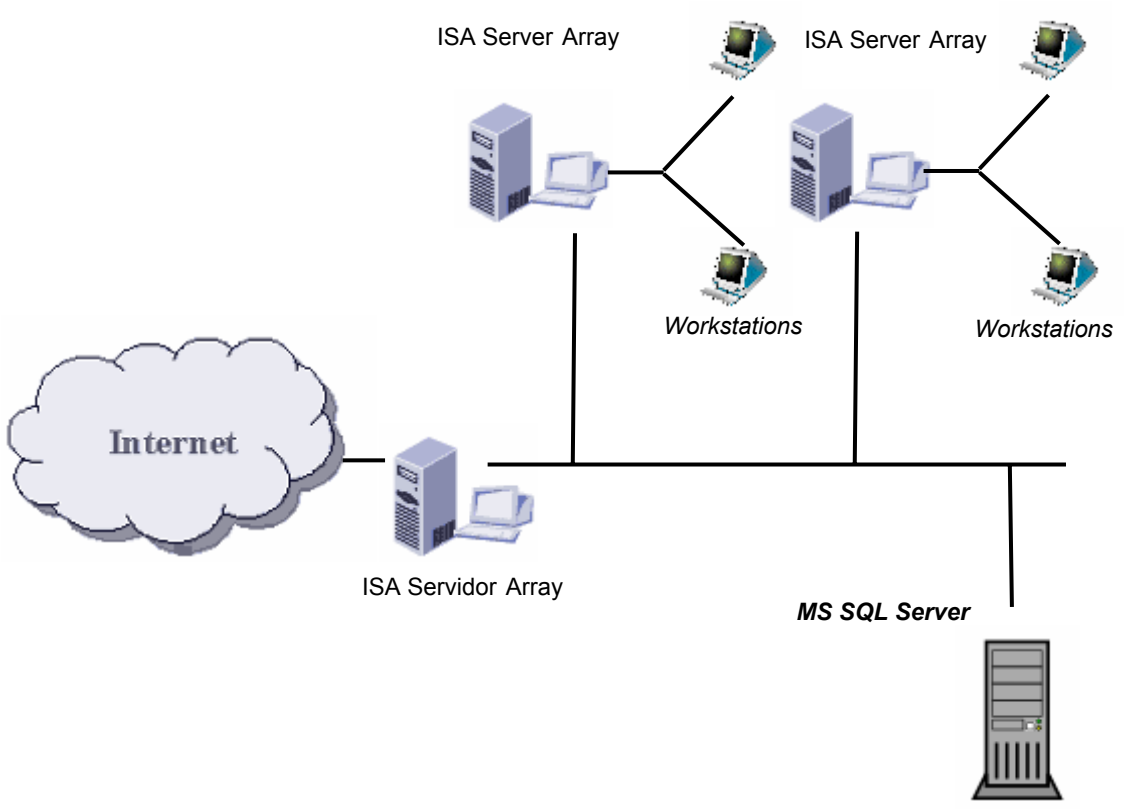


Ilustración 14. Modelo de servidores ISA Server Enterprise en *Array*

6.1.1.2 Servidores independientes

El segundo tipo de arquitectura ISA es ISA como servidor independiente, que se caracteriza por:

- Está limitada únicamente a un servidor
- No tiene por qué ser miembro de un dominio de Windows a diferencia de los servidores *array*.
- No se implementan políticas de empresa, solamente se implementan políticas de servidor.

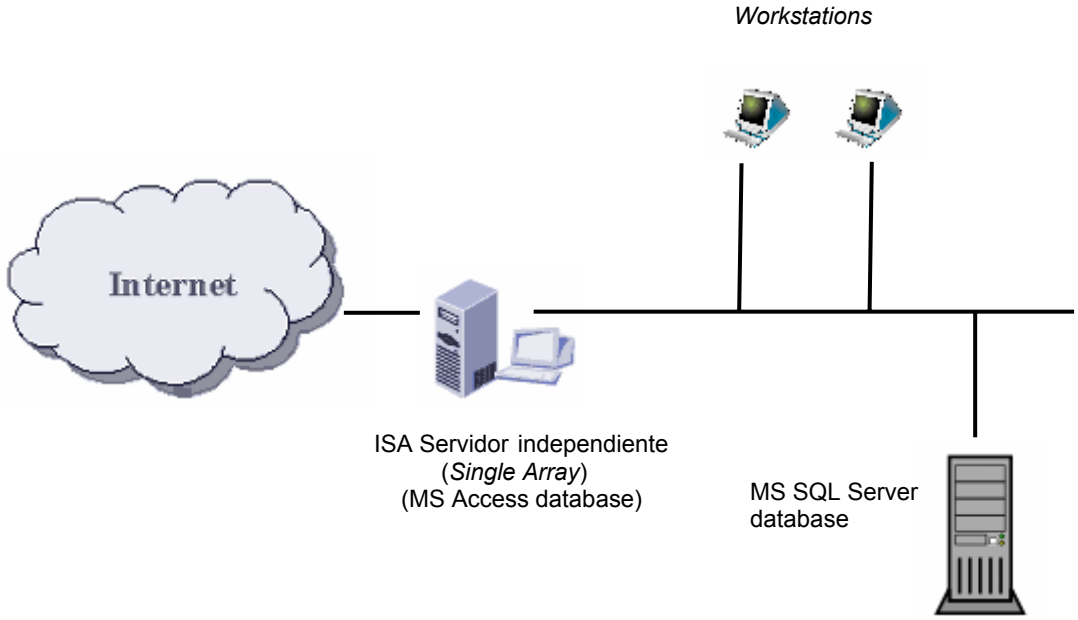


Ilustración 15. Modelo de servidor ISA Server independiente (*single Array*)

6.2 Configuración del *Firewall*

La configuración con ISA Server es con interfaz gráfica lo que ayuda a crear las reglas de manera más clara y pautada.

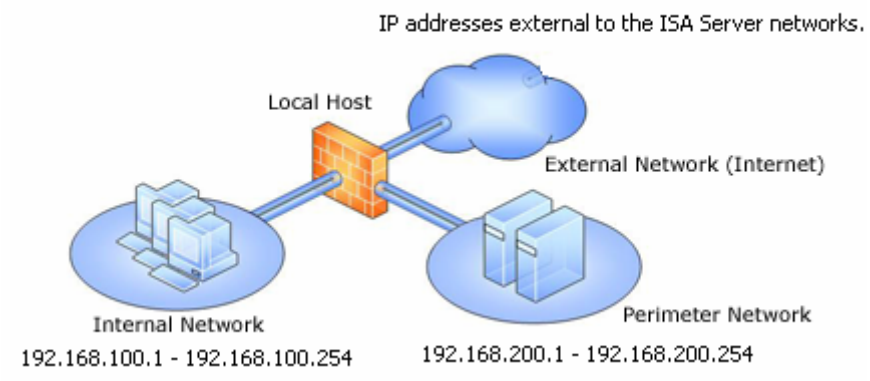


Ilustración 16. Escenario implementado en ISA Server

La política utilizada por defecto de ISA Server es de denegar todo tipo de tráfico, por eso nosotros habilitamos el tráfico que nos interese.

En primer lugar definimos el escenario *perimétrica* de nuestra red junto con los rangos de las distintas subredes.

A continuación creamos las reglas. ISA Server permite configurar las reglas de dos modos correlacionados entre sí:

- Publicación de un servidor
- Accesos a un *array*

6.2.1 Publicación de un servidor

Esta regla permite que los usuarios de una red externa puedan acceder a través del natting, que lo ejecuta esta misma regla implícitamente, al servidor que se publica en la red interna o en la DMZ.

Cuando creamos una regla de publicar un servidor, no sólo habilitamos el servidor que está detrás del *Firewall* sino que podemos filtrar los paquetes según el protocolo (*filtering*).

Para cada uno de los servicios de la DMZ creamos una regla de publicación que viene definida por los siguientes parámetros:

- General
- Acción
- Desde
- Hacia
- Tráfico
- Horas
- Redes *Listener*
- Nombre público
- *Bridging*
- Usuarios

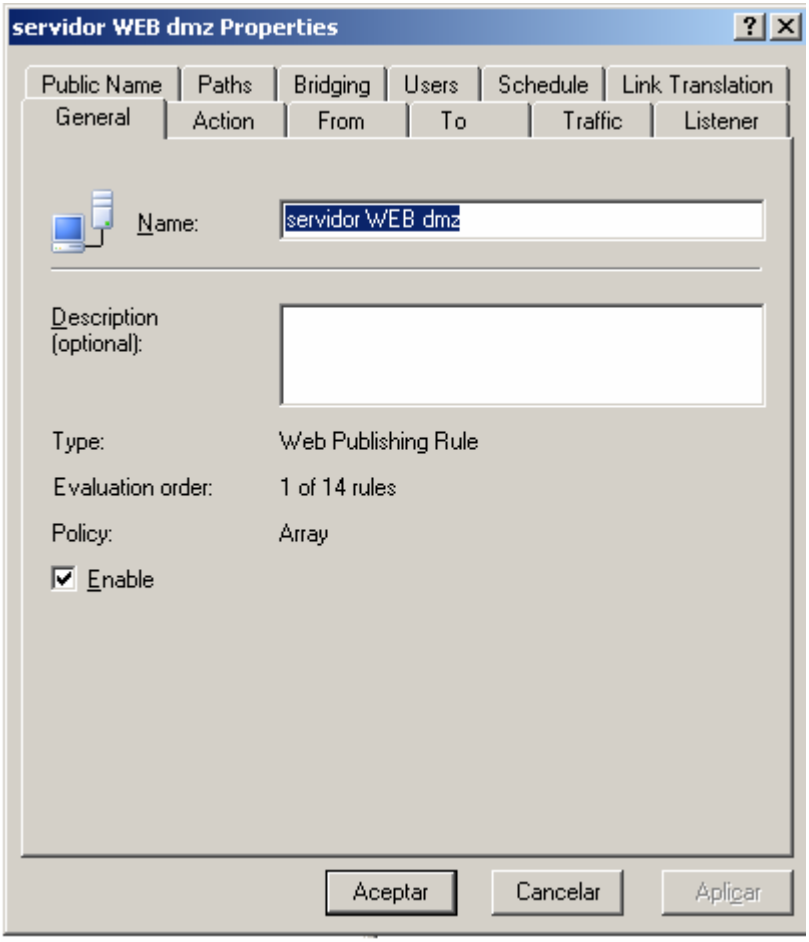


Ilustración 17. Publicación de un servidor con ISA Server

Para el caso del Nombre público, *Bridging* y Usuarios es únicamente para los servidores Web y FTP.

Cabe decir que con el parámetro *Bridging* definimos el tipo de "Servidor Web" que habilitamos ("www o Ftp"), y es aquí donde les especificamos el puerto en el que se redireccionarán las peticiones.
Mediante el *Bridging* HTTPS se puede cifrar:

- Cifrado entre cliente-*firewall* y *firewall*-servidor
- Cifrado entre cliente-*firewall*
- Cifrado entre *firewall*-servidor

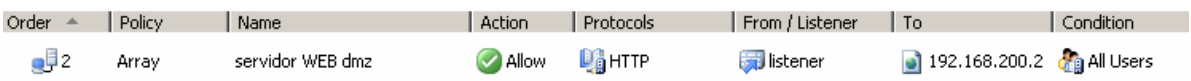


Ilustración 18. Regla publicación de un servidor con ISA Server

En el Anexo II-C.2 podemos ver en forma esquemática la descripción de cada uno de los parámetros según el servicio al que hacen referencia.

6.2.2 Accesos a *los arrays*

Además de publicar los servicios que hay detrás del *Firewall*, debemos crear las reglas para habilitar los puertos pertinentes de los servicios en cada una de las subredes.

Estas reglas que filtran a nivel de red y aplicación utilizan un sistema encadenado, es decir la primera que se cumpla es la que se ejecuta.

La configuración de cada uno de los servicios se basa en los siguientes parámetros:

- General
- Acción
- Tráfico
- Desde
- Hacia
- Horas
- Protocolos
- Usuarios
- Contenido

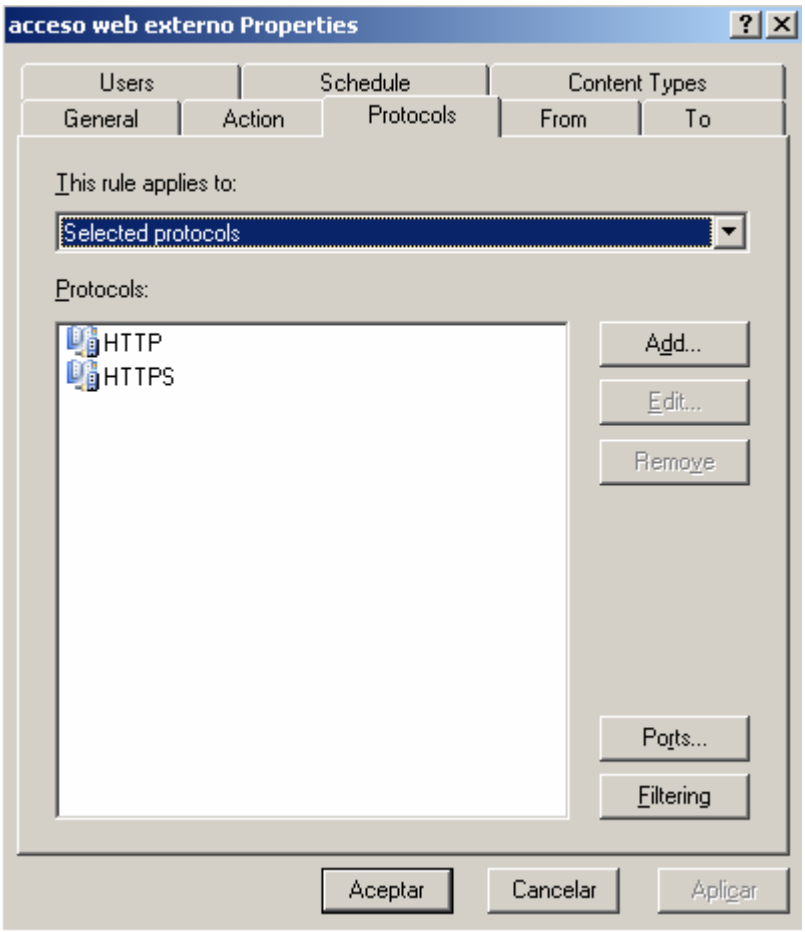


Ilustración 19. Acceso a un array con ISA Server

Como en el caso anterior, podemos ver de manera esquemática las reglas descritas (véase Anexo II-C.2).

6.3 Pruebas de vulnerabilidad

Una vez comprobado el correcto funcionamiento del *Firewall* nos disponemos a hacer un escaneo de vulnerabilidades con Nessus en los servicios de la DMZ.

En base al informe de resultados que nos proporciona Nessus nos disponemos a explicarlos del mismo modo que con el *Firewall* de *Iptables*.

Tabla de los servicios del host (DMZ)

Los servicios que están habilitados por el *Firewall* ISA Server son los que detecta Nessus y los lista junto con el tipo de mensajes que tiene sobre cada uno.

Análisis del *Host*	
Puerto/Servicio	**Mensaje respecto al puerto**
general/tcp	Anotaciones de seguridad
ftp (21/tcp)	Avisos de seguridad
ssh (22/tcp)	Notas de seguridad
smtp (25/tcp)	Notas de seguridad
domain (53/tcp)	Notas de seguridad
www (80/tcp)	Notas de seguridad
domain (53/udp)	Notas de seguridad
general/udp	Notas de seguridad

Tabla 8. Análisis del escaneo de puertos de la DMZ con ISA Server

Gráfica del tráfico en la red

En esta gráfica adjunta podemos ver el tráfico que generan estos servicios.

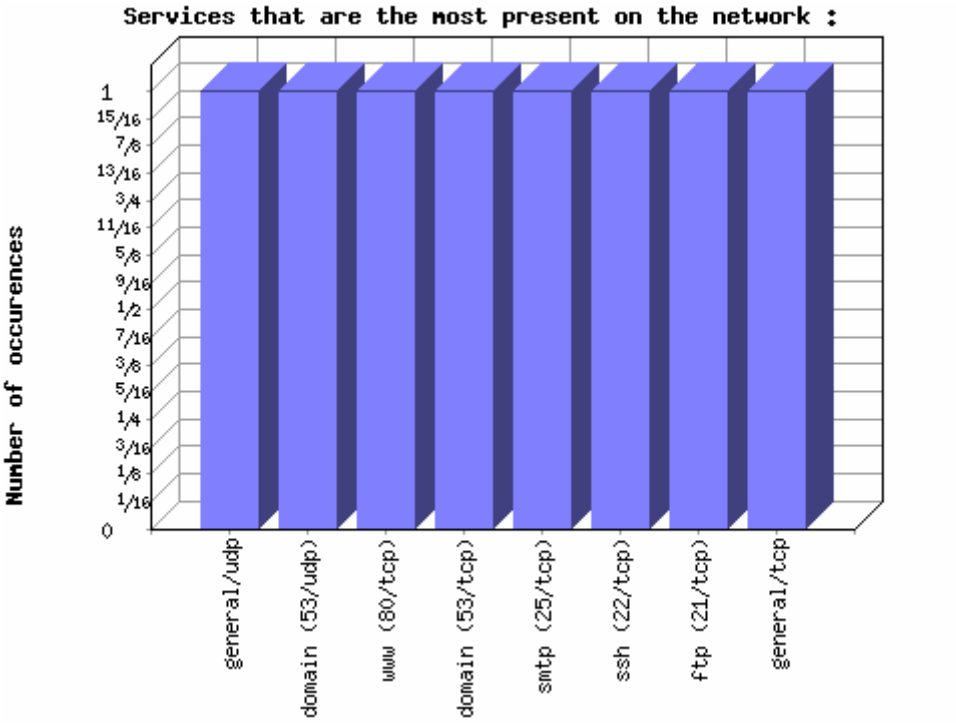


Ilustración 20. Análisis del tráfico de la red perimetral con ISA Server

Gráfica de los riesgos de seguridad

Como podemos ver en la Ilustración 17 el porcentaje de alto riesgo es nulo, o sea que Nessus no detecta ningún agujero de seguridad gracias a la protección de ISA Server.

Además, el grado de riesgo medio es muy bajo (17%) respecto el grado de riesgo bajo (83%).

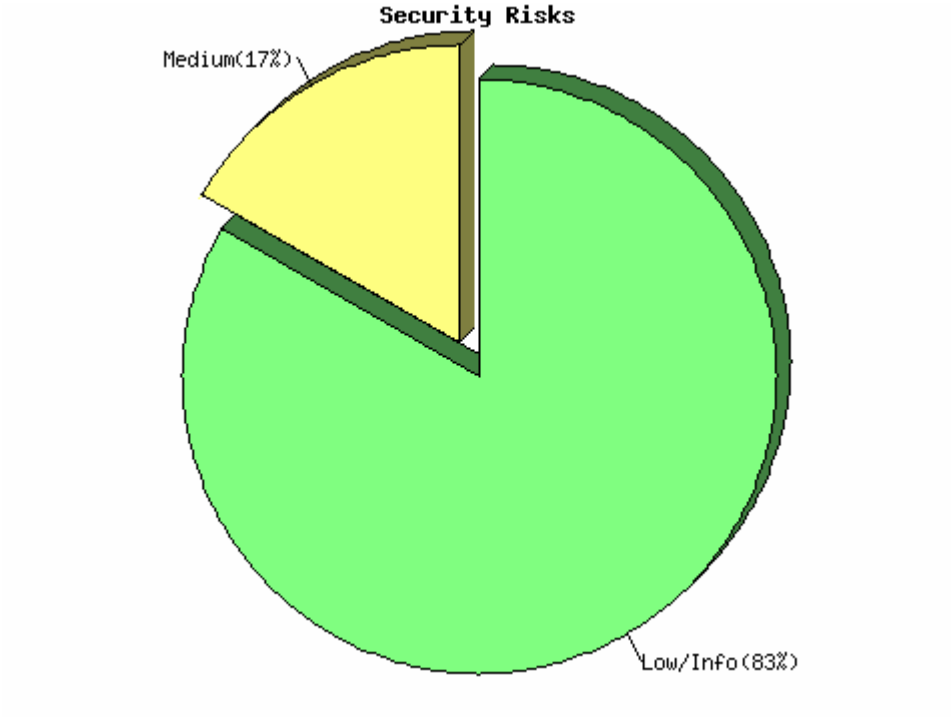


Ilustración 21. Gráfica del porcentaje de riesgo en los servicios con ISA Server

Vulnerabilidades en los servicios

En base a la tabla adjunta en el Anexo III-B podemos sacar los siguientes informes de vulnerabilidad para cada uno de los servicios:

Servidor Web

Mensaje	Contenido	Problema	Solución
Notas	Habilitados métodos TRACE/TRACK (rastreo y localización)	Permite que el atacante envíe un paquete de rastreo y si el servidor lo tiene habilitado le responde y permitirá que éste lea la ***cookie***	Añadimos en el fichero de configuración del Apache /etc/apache/httpd.conf: RewriteEngine on RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK) RewriteRule .* - [F]
	Apache/1.3.33 (Debian GNU/Linux)	Con un simple telnet al puerto 80 el servidor responde dando su versión	Cambiamos la opción del servidor Apache 'ServerTokens Prod' ServerTokens min' Así dejamos de publicar información sobre el servidor utilizado.
	Servidor web en puerto 80	-	-
	Servidor proxy en puerto 80	-	-
Aviso	/cgi-bin, /icons, /images	Debemos asegurarnos que pertenecen a un estándar de seguridad	Estos archivos que se encuentran en el directorio /usr/lib/ y /usr/share y que muestran que el servicio está abalado un estándar de seguridad, Mailman (software GNU).
Bug	-	-	-

Tabla 9. Vulnerabilidades en el servidor WEB con ISA Server

Servidor de Correo SMTP

En este caso, Nessus no detecta ningún agujero de seguridad.

Mensaje	Contenido	Problema	Solución
Notas	Servidor de correo SMTP en puerto 25	-	-

	Smtpscan no es capaz de identificar al servidor. Puede ser: Sendmail 8.11.0/8.9.3 Sendmail 8.11.6 Sendmail 8.12.3 Sendmail 8.12.11/8.12.11 Sendmail Switch-2.2.0 Sendmail 8.11.6p2/8.11.6 -112- Sendmail 8.12.8/8.11.6 -72- Postfix 2.0.7 Sendmail 8.11.6/8.11.6 -34- Sendmail 8.12.6/8.12.6 Sendmail 8.11.6+Sun/8.9.3 Sendmail 8.11.6p2/8.11.6 Sendmail 8.11.0/8.11.0 Sendmail 8.12.3/8.12.3/SuSE Linux 0.6 Sendmail 8.12.8/8.11.6 -68- Sendmail 8.11.6 Sendmail 8.11.6p2/8.11.6 -145- Sendmail 8.12.2/8.12.2/SideWinder Firewall Sendmail 8.12.8/8.11.4 (2Fast Internet Services) Sendmail 8.11.6p2/8.11.6 -134		
Aviso	-	-	-
Bug	-	-	-

Tabla 10. Vulnerabilidades en el servidor de correo con ISA Server

Servidor de Secure Shell

Del mismo modo que para el *Firewall* de *Iptables* hemos añadido en una de las reglas referentes al servicio ssh, que se habilite el tráfico externo que accede al servidor SSH de la red perimetral.

Mensaje	**Contenido**	**Problema**	**Solución**
	Servidor de SSH en el puerto 22	-	-
Notas	Versión SSH Remoto: SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4 Mecanismos de autenticación soportados: publickey,keyboard-interactive	-	-

	La versión del *daemon* SSH soporta las siguientes versions de protocolo SSH: . 1.99 . 2.0	-	-
Aviso	-	-	-
Bug	-	-	-

Tabla 11. Vulnerabilidades en el servidor SSH con ISA Server

Servidor FTP

Mensaje	Contenido	Problema	Solución
Notas	Versión: ProFTPD 1.2.10 Server	-	-
	Servidor de FTP en el puerto 21	-	-
Aviso	Versión obsoleta a la 1.2.10.	Es posible ver los usuarios que se pueden conectar al servidor remoto.	Actualizar la versión
		Las directivas ftpshut y SQLShowInfo son un arma para el atacante.	
Bug	-	-	-

Tabla 12. Vulnerabilidades en el servidor FTP con ISA Server

Servidor DNS

Mensaje	Contenido	Problema	Solución
Notas	Servidor DNS en el puerto 53	-	-
	Servidor BIND. Permite el acceso remoto de usuarios para conocer la versión del *daemon* que se ejecuta, pues se almacena en 'versión.bind' La versión de BIND remoto es: 8.4.6-REL-NOESW	El atacante tenga información adicional que le pueda servir de ayuda.	Usar la directiva 'version' en el parámetro 'options' y así bloquear 'version.bind', aunque no se bloquean las peticiones.
Aviso	-	-	-
Bug	-	-	-

Tabla 13. Vulnerabilidades en el servidor DNS con ISA Server

Una vez descritas las vulnerabilidades de los servidores y habiendo corregido aquellas que pueden ser una amenaza para la seguridad del sistema podemos sacar conclusiones sobre el comportamiento de nuestro *Firewall*.

6.4 Valoración

Sistema Operativo

En este caso, no hemos tenido que estudiar el funcionamiento del Sistema Operativo sobre el que trabajamos porque Windows es la plataforma usada comúnmente a nivel doméstico, y las funcionalidades que difieren Windows Server 2003 del Windows de uso doméstico no han sido necesarias para la configuración del *Firewall*, con lo cual no hemos tenido que invertir tiempo en ello.

Política de seguridad

ISA Server tiene por defecto deshabilitados todos sus puertos. De ninguna de las maneras puede ningún tipo de tráfico atravesar el *Firewall*; pues ésta política es la más segura, con lo que es la que hemos escogido en la configuración de los *Firewalls*.

Configuración

Para empezar, debemos definir la puesta a punto de ISA Server Enterpise como una aplicación sencilla de instalar. Ésta es una aplicación ideal para ámbitos empresariales, siendo que tiene un gran abanico de posibilidades que no se aprovecharían para una pequeña red doméstica.

La configuración de las reglas es aparentemente sencilla gracias a la interfaz gráfica, pero por la misma razón que ISA Server tiene múltiples aplicaciones, también complica ligeramente su configuración.

Tipo de filtrado

Como ya hemos comentado al comienzo del capítulo, este *Firewall* tiene la posibilidad de examinar el contenido del paquete y filtrar según el protocolo de usuario, lo que hace que el filtrado sea mucho más exhaustivo a la par que sofisticado. Sin embargo, a priori, la configuración de las reglas, parece ser sencilla pero si tenemos en cuenta el gran abanico de posibilidades de filtrado que puede llevar a cabo, su configuración deja de ser trivial. Para cada una de las reglas que se configuran existe una opción para filtrar según el protocolo referente a esa regla. Como por ejemplo, la inspección de los paquetes RPC de HTTP.

Características

ISA Server como *Firewall* supervisa las solicitudes y las respuestas transmitidas entre Internet y los equipos cliente internos, y controla quién tiene acceso a qué equipos de la red corporativa. Del mismo modo que también controla a qué equipos de Internet tienen acceso los clientes internos asegurando un elevado grado de seguridad.

Una de las ventajas que caracteriza ISA Server que hemos podido comprobar es que permite la publicación segura en Internet; pues se puede utilizar ISA Server para definir una directiva de publicación, proteger los servidores de publicación internos y permitir que los clientes de Internet tengan acceso seguro a los mismos. Además hemos podido comprobar que podemos colgar un servidor en un puerto no estándar.

A más a más, esta aplicación es capaz de abrir conexiones mediante el *tunneling*. Para el caso de una empresa, cada vez más empleados trabajan desde sus casas y utilizan desde sus equipos domésticos el acceso telefónico a la red corporativa. Por ello, es también cada vez más habitual que los empleados establezcan una conexión de red privada virtual (VPN). En esta situación, el usuario llama a su ISP local. En el otro extremo, uno de los servidores de la red corporativa está conectado a su ISP y se establece un túnel entre ambos, el cual ISA Server es capaz de aceptar siempre y cuando se cumplan unas condiciones de seguridad. Este estudio no hemos podido valorarlo a la práctica lo que no descarta que lo dejemos de tener en cuenta.

Por otra parte, aunque no lo hemos desarrollado en este proyecto debemos considerar para líneas de trabajo futuro que ISA Server es capaz de implementar una caché para los objetos que se solicitan con más frecuencia. Puede configurar la caché para asegurarse de que contiene los datos utilizados con más frecuencia en una organización o los datos a los que los clientes de Internet tienen acceso con más frecuencia.

Es importante mencionar la capacidad de ISA Server de agregar hosts que conectados entre ellos se puedan sincronizar para hacer un balanceo de carga. Una empresa que se está extendiendo continuamente requiere que su sistema de seguridad sea escalable e ISA Server lo permite con el *clustering*.

Uno de los inconvenientes que tiene este *Firewall* es su elevado coste con lo cual para según que nivel de seguridad se quiera ofrecer puede o no compensar este gasto. Esta aplicación Servidor de la multinacional Microsoft cuesta 5053,32 Euros por procesador.

Calidad de servicio

ISA Server tiene la posibilidad de priorizar el tráfico en las conexiones ya mencionadas de VPN.

Análisis de vulnerabilidades

Por otra parte, en este caso no se encuentra ninguna vulnerabilidad en el servidor de correo, pues ISA no permite el envío del comando MAIL FROM.

En este punto hemos observado que ISA Server no ha cubierto el 100% de la seguridad (sin agujeros ni notificaciones ni avisos), lo cual nos ha hecho llegar a la conclusión de que al no tener implementados los servidores de Microsoft, como ***Microsoft IIS*** de servidor Web o Microsoft Exchange de servidor de correo, ISA Server no cubre las garantías de seguridad ante los ***EXPLOITS*** de Nessus de la misma manera.

A modo de conclusión

En definitiva ISA Server ofrece un gran abanico de posibilidades que ayudan a poder decir que las garantías de seguridad se cumplen a muy alto nivel. Además de que tenemos que tener en cuenta que ISA Server es una plataforma no sólo utilizada con funcionalidades de *Firewall*, con lo cual la hace ser muy potente.

7. *Firewall*: *Packet Filter* sobre OpenBSD 3.6

El capítulo 7 va a estar definido del mismo modo que los capítulos 5. y 6. desarrollando un estudio teórico, una configuración práctica y un análisis de vulnerabilidades del *Firewall* basado en *Packet Filte*r.
Los apartados de este capítulo serán los siguientes:

- Descripción del *Firewall* con *Packet Filter*
- Configuración de las reglas de *Firewall*
- Pruebas de vulnerabilidad
- Valoración

7.1 Descripción del *Firewall* con Packet Filter

Packet Filter (PF) es el paquete de filtrado de OpenBSD. Los criterios que usa *Packet Filter* para inspeccionar los paquetes los toma de la información existente en el nivel 3 y 4 de la torre OSI o sea nivel de red (IP) y transporte (TCP, UDP, ICMP) respectivamente, como también filtra según las cabeceras de los paquetes. Los criterios que más se utilizan son los de la dirección de origen y de destino, el puerto de origen y de destino, y el protocolo.

El *Firewall* se configura en base a unas reglas de filtrado que especifican los criterios con los que debe concordar un paquete y la acción a seguir, bien sea bloquearlo o permitir su salida. Estas reglas se evalúan por orden de secuencia.

7.1.1 Sintaxis de las reglas

La sintaxis general para las reglas de filtrado es la siguiente:

```
action direction [log] [quick] on interfaz [af] [proto protocol] \
   from src_addr [port src_port] to dst_addr [port dst_port] \
   [tcp_flags] [state]
```

A continuación explicaremos los distintos parámetros de la regla:

- *action*

La acción a seguir para los paquetes que concuerden, ya sea `pass` o `block`. La acción `pass` permitirá el paso al paquete de vuelta hasta el núcleo del sistema, para que éste lo procese, mientras que la acción `block` actuará según se indique en la configuración de la opción de la política de bloqueo, `block-policy`. La acción predeterminada se puede anular especificando `block drop` (bloquear y eliminar el paquete) o `block return` (bloquear y devolver el paquete).

- *direction*

La dirección en la que se mueve el paquete en una interfaz, que será `in` (entrante) o `out` (saliente).

- `log`

Indica que se debe registrar el paquete. Si la regla especifica la opción `keep state`, `modulate state`, o `synproxy state` entonces sólo se registrará el paquete que establezca el estado. Para registrar todos los paquetes hay que usar la opción `log-all`.

- `quick`

Si un paquete concuerda con una regla que especifique la opción `quick`, entonces esa regla se considera como la regla final de concordancia para el paquete, y se tomará la acción que esté especificada en *`action`*.

- `interfaz`

El nombre o el grupo de la interfaz de red a través del cual se mueve el paquete.

- *af*

La familia de direcciones del paquete, que será `inet` para IPv4 ó `inet6` para IPv6. Generalmente, PF es capaz de determinar este parámetro basándose en la dirección, o direcciones, de origen y/o de destino.

- *protocol*

El protocolo de la capa 'Layer 4' del paquete:

  - TCP
  - UDP
  - ICMP
  - ICMP6

Un nombre de protocolo válido del fichero `/etc/protocols`
Un número de protocolo entre 0 y 255
Un grupo de protocolos que usen una lista.

- *src_addr*, *dst_addr*

La dirección de origen y/o de destino en la cabecera IP. Las direcciones se pueden especificar como:

- *src_port*, *dst_port*

El puerto de origen y/o de destino en la capa 'Layer 4' de la cabecera IP. Los puertos se pueden especificar como:

  - Un número entre el 1 y el 65535
  - Un nombre de servicio válido del fichero `/etc/services`
  - Un grupo de puertos que usen una lista
  - Un indicador de campo (<,=,>, etc)

- *tcp_flags*

Especifica los indicadores que deben existir en la cabecera TCP cuando se usa `proto tcp`. Los indicadores se especifican como `flags` *check/mask*. Por ejemplo, `flags S/SA` instruye a PF para que sólo mire los indicadores S y A (SYN y ACK), y que acepte la concordancia si el indicador SYN está activo ("on").

- *state*

Especifica si se guarda la información sobre el estado en paquetes que concuerden con esta regla.

7.1.2 Calidad de Servicio (QoS)

En la versión 3.0 de OpenBSD, se ha implementado la plataforma ***ALTQ*** (*Alternate Queueing*) de manera que encontramos integrado en la propia herramienta de *Packet Filter* la posibilidad de priorizar el ancho de banda.

La implementación de ALTQ de OpenBSD tiene soporte para ***schedulers*** de Colas Basadas en Clase (CBQ) y Colas Basadas en Prioridades (PRIQ). También tiene soporte para Pronta Detección Aleatoria (RED) y Notificación Explícita de Congestión (ECN). Por defecto, OpenBSD utiliza un *scheduler* FIFO (*First in, First Out*).

Siendo que se desvía del tema principal de este proyecto, nos disponemos a explicar brevemente y a groso modo en qué consisten estos *schedulers* para tener una visión general, por si en líneas de trabajo futuro se decide trabajar con *Packet Filter* y QoS.

First In-First Out (FIFO)
Algoritmo que consiste en generar una cola de manera que el primer paquete que entra es el primero de la cola en salir, y así sucesivamente para el resto de paquetes.

Colas Basadas en Clase (CBQ)
CBQ (*Class Based Queueing*) es un algoritmo de formación de colas que divide el ancho de banda de una conexión de red entre varias colas o clases. A cada cola se le asigna un tráfico basándose en la dirección de origen o de destino, el número de puerto, protocolo, etcétera.

Colas Basadas en Prioridades (PRIQ)
Las PRIQ (*Priority Queueing*) asignan colas múltiples a una interfaz de red, y dan a cada cola un nivel de prioridad único. Una cola con un nivel de prioridad más alto se procesa *siempre* antes que una cola con un nivel de prioridad más bajo.

Pronta Detección Aleatoria (RED)

RED (*Random Early Detection*) es un algoritmo que se utiliza para evitar la congestión. Su trabajo es evitar la congestión en la red, asegurándose de que la cola no se llene.

7.2 Configuración de las reglas del *Firewall*

Para empezar, antes de crear las reglas por las que se regirá el *Firewall* debemos activar la opción de enrutamiento y así la máquina sea capa de redireccionar el tráfico de las distintas redes. Para ello nos editamos el fichero `/etc/sysctl.conf`

```
net.inet.ip.forwarding=1  #1=Permit
```

Además habilitamos en el archivo de la configuración del sistema localhost `/etc/rc.conf` el paquete de *Packet Filter*, valga la redundancia.

```
pf=YES
```

Una vez habilitados estos parámetros configuramos las reglas del *Firewall* editando el fichero `/etc/pf.conf` (véase Anexo II-C.3).

A continuación, para cargar el archivo pf.conf debemos hacerlo con la utilidad `pfctl`. Si consultamos su manual (`man pfctl`) vemos que debemos utilizar el comando:

```
pfctl –f /etc/pf.conf
```

7.3 Pruebas de vulnerabilidad

De la misma manera que con los *Firewalls* de *Iptables* e ISA Server verificamos el correcto funcionamiento del *Firewall* antes de disponernos al estudio de vulnerabilidades.

Una vez hecha la comprobación testeamos los servicios de la DMZ con la aplicación Nessus.

Tabla de los servicios del *host* (DMZ)

Los servicios que están habilitados por el *Firewall* de *Packet Filter* son los que detecta Nessus y los lista junto con el tipo de mensajes que tiene sobre cada uno.

	Análisis del *Host*
Puerto/Servicio	**Mensaje según el puerto**
general/tcp	Notas de seguridad
ftp (21/tcp)	Avisos de seguridad
ssh (22/tcp)	Notas de seguridad
smtp (25/tcp)	Agujero de seguridad
www (80/tcp)	Avisos de seguridad
pop3 (110/tcp)	Notas de seguridad
domain (53/udp)	Notas de seguridad
general/udp	Notas de seguridad

Tabla 14. Análisis del escaneo de puertos de la DMZ

Gráfica del tráfico en la red

En esta gráfica adjunta podemos ver el tráfico que generan los servicios y sus respectivos puertos (eje X) en base al número de ocurrencias (eje Y).

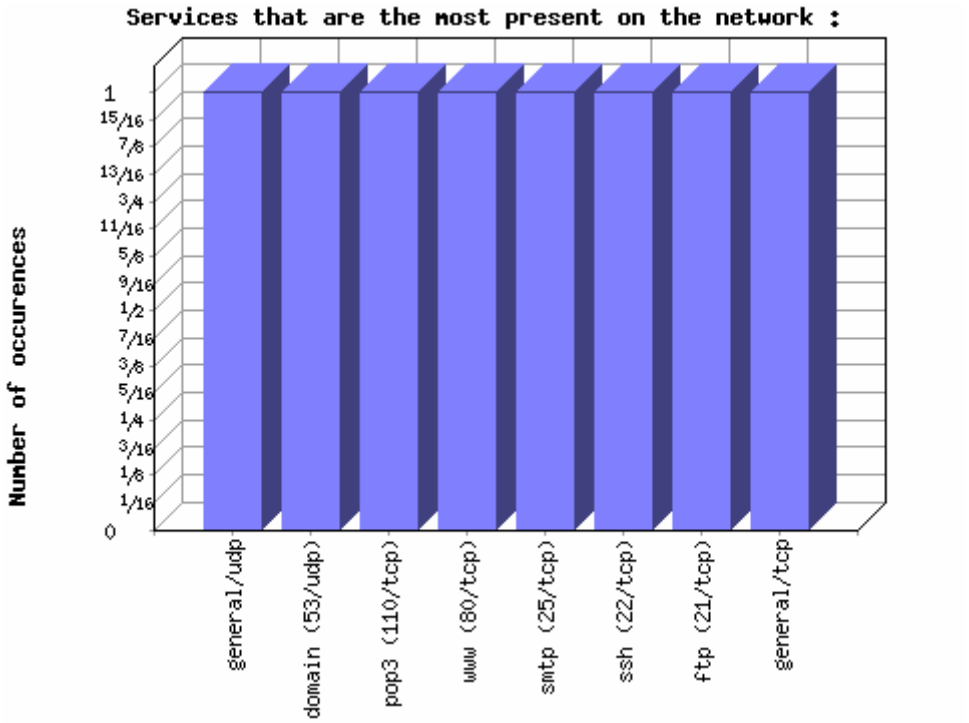


Ilustración 22. Análisis del tráfico de la red perimetral con *Packet Filter*

Gráfica de los riesgos de seguridad

Como podemos ver en la Figura 2 el porcentaje de alto riesgo (agujero de seguridad) es del 3% donde en la gráfica 3 podemos ver cual es el servicio o servicios que lo provocan (el de correo).
Por otra parte, el grado de riesgo medio es del (27%) respecto el grado de riesgo bajo (70%).

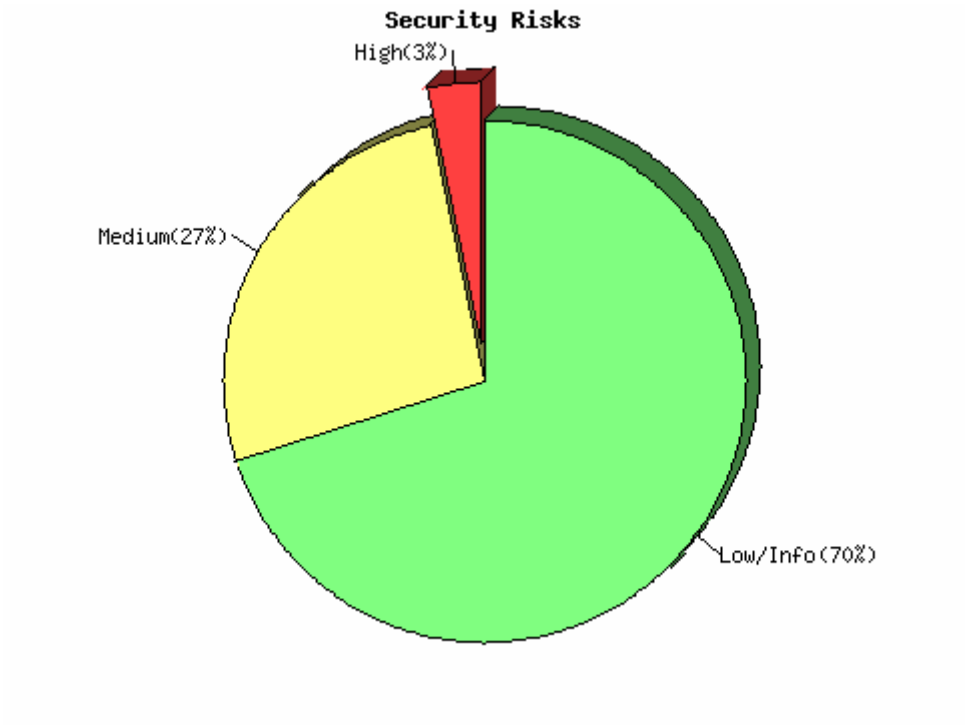


Ilustración 23. Gráfica del porcentaje de riesgo en los servicios con *Packet Filter*

Agujeros de seguridad en el escenario

Como vemos en la figura 3, según el eje de las X (nombre del servicio) y el eje de las Y (número de agujeros de seguridad), el único agujero que Nessus ha detectado se ve provocado por el servicio de correo.

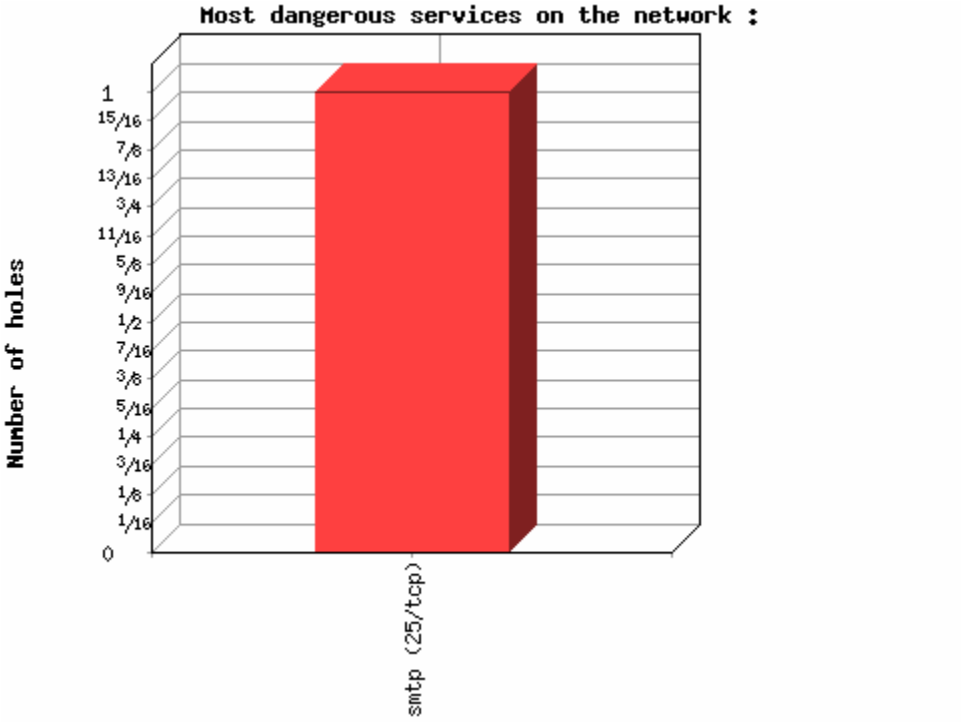


Ilustración 24. Gráfica de los servicios con agujeros de seguridad

Vulnerabilidades en los servicios

En base a la tabla adjunta en el Anexo III-C. podemos sacar los siguientes informes de vulnerabilidad para cada uno de los servicios:

Servidor Web

Mensaje	Contenido	Problema	Solución
Notificación	Existe un servidor web en este puerto	-	-
Aviso	El servidor intenta esconder su versión y su nombre, sin embargo Nessus ha sido capaz de identificarlo Apache/ 1.3.33	No hay riesgos	Fijar la configuración
Bug	-	-	-

Tabla 15. Vulnerabilidades en el servidor WEB con *Packet Filter*

Servidor de Correo SMTP

De la misma manera que en el *Firewall* de *Iptables* Nessus detecta un agujero de seguridad en el servicio de correo SMTP.

Mensaje	Contenido	Problema	Solución
Notificación	Servidor de correo SMTP en puerto 25	-	-
Aviso	El servidor SMTP responde a los comandos EXPN y VRF.	El comando EXPN puede utilizarse para encontrar los alias de direcciones de correo incluso el nombre completo del destinatario. El comando VRF se puede utilizar para comprobar si es válida una cuenta. El servidor no debe permitir que usen estos comandos usuarios remotos.	Añadir la opción en el fichero /etc/sendmail.cf O PrivacyOptions=goaway
Bug	No reacciona ante el comando: MAIL FROM |testing	Permite que cualquiera pueda usar comandos en este host	Actualización de sendmail

Tabla 16. Vulnerabilidades en el servidor de correo con *Packet Filter*

Servidor de Secure Shell

Mensaje	Contenido	Problema	Solución
Notificación	Servidor de SSH en el puerto 22	-	-
	Versión SSH Remoto: SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4 Mecanismos de autenticación soportados: publickey,keyboard-interactive	-	-

	La versión del *daemon* SSH soporta las siguientes versiones de protocolo SSH: . 1.99 . 2.0	-	-
Aviso	-	-	-
Bug	-	-	-

Tabla 17. Vulnerabilidades en el servidor SSH con *Packet Filter*

Servidor FTP

Mensaje	Contenido	Problema	Solución
Notificación	Versión: ProFTPD 1.2.10 Server	-	-
	Servidor de FTP en el puerto 21	-	-
Aviso	Versión obsoleta a la 1.2.10.	Es posible ver los usuarios que se pueden conectar al servidor remoto.	Actualizar la versión
		Las directives ftpshut y SQLShowInfo son un arma para el atacante.	
Bug	-	-	-

Tabla 18. Vulnerabilidades en el servidor FTP con *Packet Filter*

Servidor DNS

Mensaje	Contenido	Problema	Solución
Notificación	Servidor DNS en el puerto 53	-	-
	Servidor BIND. Permite el acceso remoto de usuarios para conocer la versión del *daemon* que se ejecuta, pues se almacena en 'versión.bind' La versión de BIND remoto es: 8.4.6-REL-NOESW	El atacante tenga información adicional que le pueda servir de ayuda.	Usar la directiva 'version' en el parámetro 'options' y así bloquear 'version.bind', aunque no se bloquean las peticiones.

Aviso			
Bug	-	-	-

Tabla 19. Vulnerabilidades en el servidor de FTP con *Packet Filter*

Del mismo que para el resto de servicios corregimos la vulnerabilidad aplicando la solución propuesta por Nessus y comprobamos que en el siguiente escaneo deja de presentar la notificación o aviso pertinente.

Una vez descritas las vulnerabilidades de los servidores y habiendo corregido aquellas que pueden ser una amenaza para la seguridad del sistema podemos sacar conclusiones sobre el comportamiento de nuestro *Firewall*.

7.4 Valoración

Sistema Operativo

Para configurar el *Firewall* con *Packet Filter* no es necesario tener un profundo conocimiento del sistema operativo OpenBSD, pero sí se requieren ligeras nociones.

Política de seguridad

Este *Firewall* sigue una política por defecto de aceptación de todos los paquetes en ambos sentidos. Partimos de la base en que todos los puertos se encuentran habilitados, lo que lo hace ser menos seguro. Por ello, para la configuración de nuestro *Firewall* hemos modificado la política de seguridad por defecto denegando todo tráfico entrante y saliente.

Configuración

Por lo que respecta a su configuración la podemos calificar como una configuración sencilla, pues por lo general todas las reglas siguen el mismo modelo, cambiando los parámetros TCP/IP. Para evitar una gran lista de reglas que hagan más complejo el mantenimiento del *Firewall*, hemos agrupado la habilitación de varios protocolos TCP en una sola regla.

Tipo de filtrado

Como ya hemos comentado a lo largo de este capítulo *Packet Filter* inspecciona los paquetes a nivel TCP/IP. Por eso nuestras reglas filtran en base a las direcciones origen o destino a los puertos origen y destino y no en

base a la capa de aplicación, lo que impide hacer un filtrado menos sofisticado. Esto implica que la carga de las reglas no sea elevada.

Características

Packet Filter permite el almacenamiento de los paquetes que interesen ser guardados, es decir aquellas reglas pertenecientes a paquetes especialmente sensibles. Así pues, en nuestro caso con la variable "log" obligamos que todos los paquetes que sean rechazados se nos notifiquen y se guarden en un fichero (`/var/log/pflog`). Pues en todo *Firewall* es muy importante que se pueda monitorizar el tráfico para tener controladas las acciones que se llevan a cabo.

Calidad de servicio

A más a más, otra de las características a destacar que dispone es la capacidad de ofrecer Calidad de Servicio (QoS) a través del ALTQ integrado en el mismo *Packet Filter*. Éste permite modelar las colas en las que los paquetes son almacenados al llegar al *Firewall*, de forma que en vez de establecer un patrón FIFO (*First in, First out*) podamos establecer reglas y prioridades en base a los orígenes, destinos y tipo de paquetes que se están tratando en cada momento.
De este modo, se evitan los retrasos en determinados servicios que son más sensibles a ello, como por ejemplo SSH.

Análisis de vulnerabilidades

Con el analizador Nessus hemos visto que no se cubre al 100% la seguridad exigida puesto que se ha encontrado un agujero de seguridad en el servidor de correo.
En el resto de servicios existen notificaciones pero sin ninguna relevancia porque las hemos podido suplir.

A modo de conclusión

El *Firewall* configurado con *Packet Filter* es poco granulado ya que los parámetros de filtrado no van más allá de la cabecera IP pero sí que es muy válido para la protección de cualquier red que no sea necesaria una seguridad tan exhaustiva. Además de que tiene la característica de poder ofrecer QoS que para cualquier empresa es necesario, ya que así se puede dar prioridad a determinado tráfico.

Una seguridad no tan sofisticada no descarta que no se pueda garantizar una seguridad al 100% dentro de sus límites; por eso se puede considerar *Packet Filter* un buen *Firewall* para cualquier organización.

8. BALANCE EN BASE AL ESTUDIO PRÁCTICO

Una vez hecho el estudio y análisis de cada uno de los tres *Firewalls* de manera independiente, podemos hacer un balance y comparar los tres para extrapolar conclusiones a un nivel más genérico.

A continuación, el capítulo constará de un segundo punto donde nos disponemos a valorar cuál de los *Firewalls* es más apropiado en según qué escenarios.

8.1 Análisis comparativo de los tres *Firewalls*

Sistema Operativo

En primer lugar debemos tener en cuenta los Sistemas Operativos sobre los que configuramos cada uno de los *Firewalls*.

La distribución Debian Sarge de Linux y OpenBSD son dos sistemas operativos UNIX, ambos desarrollados por proyectos no comerciales. Una de las diferencias más importantes entre estos dos sistemas operativos es que los sistemas BSD pueden ejecutar código de Linux, mientras que Linux no puede ejecutar código de BSD, lo que provoca que haya una mayor cantidad de software para BSD que para Linux.

Por otra parte, Windows 2003 Server pertenece a una entidad privada (Microsoft), utilizada por la inmensa mayoría de usuarios. Así pues facilita el propio manejo de cualquier aplicación que trabaje sobre Windows, pero eleva muy considerablemente el coste del equipo. (Véase 0 Análisis de costes). No obstante, el hecho de que la plataforma Windows sea la más utilizada actualmente es también la que está más expuesta a virus y a ataques.

Política de seguridad

En segundo lugar, debemos considerar las políticas por defecto de cada uno de los *Firewalls*, pues hemos podido comprobar que tanto para *Iptables* como para *Packet Filter* se admiten de partida todo tipo de paquetes entrantes y salientes. Sin embargo, ISA Server estipula por defecto una política de negación de tráfico, lo que a priori garantiza mayor seguridad.

Configuración

Por lo que respecta a la configuración de las reglas, la que ha resultado ser más sencilla es con *Iptables*. El inconveniente es que se debe configurar una regla por cada tipo de tráfico que deseamos habilitar, con lo cual la lista de

reglas que acaba teniendo el *Firewall* es bastante considerable, lo que dificulta su mantenimiento.
Para el caso de *Packet Filter*, la configuración no es muy complicada pero si requiere tener cierto dominio de la herramienta ya que no es tan lógica como Iptables.

En cambio, para el caso de ISA Server, aparentemente parece una configuración sencilla, pero si se quieren tener en cuenta la infinidad de posibilidades que ofrece la aplicación, la configuración no es tan trivial.

Tipo de filtrado

Dado el estudio de los tres *Firewalls* hemos podido ver que ISA Server 2004 Enterprise (como *Firewall*) es mucho más completo que los otros dos por una serie de elementos que lo distinguen:

- Para empezar ISA Server tiene unas funcionalidades de filtrado mucho más granuladas y sofisticadas que *Packet Filter* e *Iptables*, pues el tráfico se filtra a nivel de aplicación mientras que los otros dos se hace a nivel de red. Esto supone que el tiempo de carga de las reglas en *Packet Filter* e Iptables no sea muy elevado.

- Sin embargo, en este caso existe una pequeña diferencia entre *Packet Filter* e *Iptables* y es que éste último permite cargar un módulo de Kernel llamado *string matching match* que permite una inspección limitada del *payload* del paquete impidiendo la entrada de ciertas peticiones al servidor Web que no interesan. Este módulo amplía ligeramente las fronteras de *Packet Filter* pero no llega a la cima para inspeccionar los paquetes más allá de este nivel.

- En cuanto a las reglas de natting, para el caso de *Iptables* y *Packet Filter* se deben configurar a parte de manera explícita mientras que en ISA Server es suficiente con una regla de publicación de servicio porque ésta ya integra las funcionalidades de *natting* y *filtering*.

Calidad de servicio

Otro punto que se debe considerar es la Calidad de Servicio (QoS). En este caso, *Iptables* carece el poder priorizar el tráfico, sólo se permite marcar los paquetes con la tabla mangle. Luego, para que se puedan establecer distintas colas según el tráfico que se ha marcado, es necesario una segunda herramienta proporcionada por el paquete Iproute2 donde se añade un control de tráfico TCP/IP, lo que no ha sido objeto de estudio en este proyecto.

Sin embargo, como se ha comentado en la valoración del *Firewall* con *Packet Filter*, es posible gestionar el ancho de banda a través del ALTQ.

La Calidad de Servicio es importante tener en cuenta ya que según qué tipo de servicios requieren que el retraso sea mínimo, con lo cual el poder priorizar un

determinado tráfico es fundamental en los casos en el que el ancho de banda es limitado.

Escalabilidad

Por lo que respecta a la escalabilidad de los *Firewalls* el tener que generar una regla por cada tipo de tráfico (en el caso de *Iptables*) provoca que al final de la configuración del *Firewall* haya un cúmulo de reglas que dificultan la gestión y el mantenimiento en el caso que se quiera ampliar la red. Este tipo de *Firewalls* adolecen de problemas de escalabilidad. En cambio ISA Server permite la agregación de hosts conectados entre ellos y su sincronización, de manera que se puede generar un balanceo de carga. Por lo que respecta a *Packet Filter* el caso es similar que *Iptables* pero en menor grado, puesto que podemos agrupar un número de reglas con distintos protocolos en una sola.

Características

Otra de las características que nos ofrece la plataforma ISA Server que no disponen *Iptables* y *Packet Filter* es el permitir tener acceso remoto seguro a la red corporativa mediante VPN's;

- ISA Server 2004 permite proporcionar acceso remoto seguro basado en estándares para conectar las sucursales y los usuarios remotos a las redes corporativas. Además da la posibilidad de ofrecer prioridades en el ancho de banda de manera que ofrece calidad de servicio (QoS).

Es importante destacar la posibilidad de que para ISA Server se pueden publicar servidores en puertos no estándares, con lo cual si un puerto estándar de determinado servicio está deshabilitado se podría colgar ese mismo servicio en el puerto 80 por ejemplo. Esto es una mejora que no disponen los otros dos *Firewalls*.

Análisis de vulnerabilidades

En cuanto al análisis de vulnerabilidades, hemos podido corroborar que ninguno de los tres *Firewalls* cumple una garantía de seguridad al 100%, pero cabe decir que los avisos que genera Nessus al respecto son irrelevantes, puesto que la mayoría se pueden corregir de manera trivial. Únicamente hay que considerar con mayor detenimiento el agujero encontrado en el servidor de correo, pues existe tanto en el *Firewall* de *Iptables* como en el de *Packet Filter* . Según estas pruebas hemos podido comprobar que difícilmente se pueden sacar diferencias entre los tres *Firewalls* porque los resultados son muy parecidos. Si bien esperábamos que esto ocurriese con los *Firewalls Iptables* y *Packet Filter* _ambos tienen unas funcionalidades de filtrado muy similares y hemos seguido un mismo modelo de configuración para los dos_ nos ha sorprendido que también sucediera con ISA Server. Ante esto, hemos llegado a la conclusión que si hubiéramos configurado los servidores de Microsoft, como son Microsoft IIS de servidor Web o Microsoft Exchange de servidor de correo, ISA Server hubiera cubierto al 100% las garantías de seguridad ante los EXPLOITS de Nessus.

Llegados a este punto sería interesante de cara a líneas de trabajo futuras un seguimiento rehacer estas mismas pruebas con dichos servidores.

A modo de conclusión

Tras los tests realizados los *Firewalls* se revelan como pieza fundamental para garantizar un alto nivel de seguridad en la red. Es indudable que Iptables o *Packet Filter* tienen carencias que ISA Server cubre pero todos ellos aseguran una protección mínima que en algunos casos ya es suficiente.

8.2 Adecuación de los *Firewalls* según el escenario empresarial

Empresa con sedes por toda España

En este caso el *Firewall* más adecuado es sin lugar a dudas ISA Server Enterprise porque en este caso se pueden aprovechar muchas de las funcionalidades de la plataforma.

Para empezar sería necesaria la configuración de VPN´s para que usuarios de otras sedes se conecten de manera remota a un punto concreto de la red de la empresa.

Además de ser fundamental poder ofrecer calidad de servicio en el tráfico priorizando aquel tráfico que proviene de una jerarquía más alta por ejemplo, o bien para priorizar aquel en el que el retraso es crítico para la buena comunicación.

En este caso, suponiendo que la red es extensa también sería adecuado aprovechar la posibilidad de ISA Server como Caché de manera que haga disminuir tráfico innecesario en la red.

Otra de las ventajas que se podría aprovechar en este caso es la posibilidad de hacer *clustering* y balanceo de carga, siempre y cuando la empresa extienda sus sedes y se fuerce la agregación de un nuevo host.

El único inconveniente que pudiera haber es el coste de la aplicación, que de alguna manera si es una gran empresa puede compensar y más si tenemos en cuenta que el mantenimiento no tiene porque suponer un gran gasto. Si la empresa no fuerza a tener grandes cambios, las pequeñas modificaciones que se tuvieran que hacer no obligarían a tener que pagar el desplazamiento de un técnico, pues como hemos dicho en la valoración de ISA Server, la propia configuración de las reglas es más trivial que la del resto de *Firewalls*.

Pequeña empresa

Para un escenario de este tipo, una pequeña red con una sola sede en una misma oficina no compensa el tener que pagar tanto por el *Firewall* si muchas de las facilidades que se ofrecen no se utilizarán al completo. Es una inversión inefectiva, puesto que un *Firewall* del tipo *Packet Filter* bien configurado será mucho más útil y eficaz. Podrá garantizar un elevado nivel de seguridad con menor coste.

Además también se podrá ofrecer Calidad de servicio si fuera necesario.

Por otra parte el mantenimiento sí que resultaría ser costoso si tenemos en cuenta que este sistema de seguridad requiere ciertas nociones que no cualquiera podría hacer sin una ayuda. Así pues, por cualquier cambio sí que se verá la empresa obligada a pagar un técnico.

9. CONCLUSIONES

Este capítulo corresponde a las conclusiones que lo conforman los siguientes puntos:

- Revisión de los objetivos
- Revisión de la planificación y los costes
- Líneas de trabajo futuro
- Conclusiones personales

Es importante tener en cuenta lo que se consideró al empezar cualquier proyecto para luego revisar si se ha cumplido.
Además de proponer una líneas de trabajo futuro de manera que todo este trabajo sirva de base para un emprender un nuevo tema.
Finalmente el último apartado corresponde a las conclusiones personales.

9.1 Revisión de objetivos

En todo proyecto es fundamental hacer un análisis sobre el cumplimiento o no de los objetivos propuestos en su comienzo.

A medida que hemos ido trabajando hemos intentado tener siempre presente los objetivos de este proyecto puesto que cuanto más profundizamos en algo en concreto más se abre el abanico de temas que se pueden considerar, pero a veces es difícil saber acotar; pues un tema tan extenso como la Seguridad en Redes, *Firewalls* , el cual ofrece un gran número de fuentes de información, ralentiza la elaboración de la memoria y no permite seguir con alguno de los objetivos marcados.

Si revisamos el apartado 1.2 donde aparecen la lista de objetivos a cumplir vemos que hemos cumplido todos ellos a excepción de la configuración y test de los *Firewalls* de más de un escenario.

Los motivos que han llevado al no poder cumplir los objetivos se describen en el apartado 9.4 (Conclusiones personales).

A continuación adjuntamos una tabla donde listamos los objetivos propuestos en un comienzo y definimos si han sido o no cumplidos.

Objetivo	**Alcanzado**
Configuración de los servicios del escenario (DNS, SSH, Correo, Web, WINS, DHCP).	Sí exceptuando el servidor WINS
Estudio de la taxonomía de *Firewalls*.	Sí
Análisis en profundidad a nivel teórico de los tres *Firewalls* con Iptables, ISA Server 2004 Enterprise y *Packet Filter*.	Sí
Configuración de los tres *Firewalls* en base a una política de negación de servicio	
Simulación de ataques y análisis de vulnerabilidades mediante el escaneo de puertos con la herramienta Nessus.	Sí
Configuración para distintos escenarios: - Escenario con DMZ. - Escenario sin DMZ - Escenarios en los que combinamos los servicios entre la DMZ y la Interna. - Valorar cada escenario y hacer una comparativa	Sí No No No
Redacción de esta memoria escrita con objeto de documentar las conclusiones alcanzadas tras cumplir los anteriores objetivos.	Sí

Tabla 20. Tabla de los objetivos propuestos y alcanzados.

9.2 Revisión de la planificación y costes

9.2.1 Planificación temporal

No es una tarea fácil el poder cumplir rigurosamente los objetivos propuestos en las fechas previstas. En la teoría, aún dejando cierto margen para posibles problemas o imprevistos que puedan aparecer en la práctica, se dejan de

prever cosas que con mayor o menor importancia pueden entorpecer la labor del proyecto. Hasta que no se adentra a fondo no se puede predecir de manera exacta lo que le te va a llevar; claro que también es fundamental compensar esas demoras no previstas para no seguir atrasando la labor del proyecto.

A continuación se muestra el esquema del desarrollo temporal real donde se podrá comprobar que ambos no se ajustan exactamente el uno con el otro. Las causas son muy variadas, algunas de ellas están explicadas en el apartado de conclusiones personales, y otras son las mismas que causan las demoras en cualquier proyecto de empresa dirigido a un cliente, la no rigurosa previsión.

WBS	Name	Start	Finish	Work	Duration	Slack
1	**Búsqueda y planificación del proyecto y de sus objetivos**	**Mrz 21**	**Apr 29**	**30d**	**30d**	**179d**
1.1	Búsqueda y filtrado de información	Mrz 21	Apr 29	30d	30d	179d
2	**Instalación de la partición: Debian Sarge y Windows 2003 Server e**	**Mai 2**	**Mai 19**	**14d**	**14d**	**165d**
2.1	Familiarizarse con las particiones	Mai 2	Mai 19	14d	14d	165d
2.2	Escribir introducción	Mai 16	Mai 18	3d	3d	166d
3	**Asignación de un nuevo despacho**	**Mai 19**	**Mai 19**	**1d**	**1d**	**165d**
3.1	Nueva ubicación, nuevo material	Mai 19	Mai 19	1d	1d	165d
4	**Planificación junto al tutor sobre la estructura del proyecto**	**Mai 23**	**Mai 24**	**2d**	**2d**	**162d**
4.1	Planificación personal del trabajo	Mai 23	Mai 23	1d	1d	163d
4.2	Comentar planificación con el tutor y modificaciones	Mai 23	Mai 23	1d	1d	163d
4.3	Creación de un índice	Mai 24	Mai 24	1d	1d	162d
5	**Familiarización con Debian y Windows 2003 Server**	**Mai 30**	**Jun 6**	**6d**	**6d**	**153d**
5.1	Instalación Debian Sarge y familiarización	Mai 30	Jun 1	3d	3d	156d
5.2	**Instalación Windows 2003 Server**	**Jun 2**	**Jun 6**	**3d**	**3d**	**153d**
5.2.1	Configuración servidores con Win 2003 Server	Jun 2	Jun 6	3d	3d	153d
6	**Configuración escenario DMZ**	**Jun 20**	**Jul 5**	**12d**	**12d**	**132d**
6.1	Configuración servidores y pruebas de funcionamiento Debian Sarge	Jun 20	Jul 1	10d	10d	134d
6.2	Escribir en memoria el escenario	Jul 1	Jul 5	3d	3d	132d
7	**Configuración del Firewall Iptables**	**Jul 4**	**Jul 12**	**7d**	**7d**	**127d**
7.1	Configuración de las reglas	Jul 4	Jul 6	3d	3d	131d
7.2	Análisis de vulnerabilidades	Jul 7	Jul 8	2d	2d	129d
7.3	Escribir en memoria la parte teórica y las pruebas elaboradas	Jul 11	Jul 12	2d	2d	127d
8	**Instalación ISA Server Enterprise 2004**	**Jul 13**	**Jul 27**	**11d**	**11d**	**116d**
8.1	Configuración de las reglas y familiarización a nivel teórico de ISA	Jul 13	Jul 18	4d	4d	123d
8.2	Análisis de vulnerabilidades	Jul 20	Jul 21	2d	2d	120d
8.3	Escribir en memoria la parte teórica y las pruebas realizadas	Jul 21	Jul 27	5d	5d	116d
9	**Repetición del escaneo de vulnerabilidades**	**Jul 28**	**Aug 1**	**3d**	**3d**	**113d**
9.1	Firewall Iptables	Jul 28	Jul 28	1d	1d	115d
9.2	Firewall ISA Server	Aug 1	Aug 1	1d	1d	113d
10	Seguir completando la escritura de la memoria de lo hecho hasta el momento	Aug 2	Aug 10	7d	7d	106d

Ilustración 25. Planificación temporal real del Proyecto

9.2.2 Análisis económico

Por lo que respecta al análisis económico, los precios que contemplamos en un principio difieren de la realidad, a causa de un incremento de horas invertidas en la elaboración del proyecto. Según la planificación temporal llevada a cabo hay un incremento de 400 horas, que proviene de una duración del trabajo de 174 días a una media entre 6 y 7 horas, dando un total de 1100 horas.

Además de haber tenido que utilizar un equipo más del previsto, pues hemos tenido que instalar Nessus desde una aplicación Knoppix y hemos necesitado el equipo en el que estaba la Debian. Es en éste donde supuestamente debíamos instalar OpenBSD. Por eso decidimos utilizar otra máquina donde configurar el tercer *Firewall*.

En definitiva el coste del proyecto aumenta de manera considerable, un 32% respecto al coste previsto en un primer momento, con lo cual el valor total de este proyecto es de **26.720,71 €**

A continuación, del mismo modo que en el análisis de costes previstos que hicimos en el comienzo de este proyecto, adjuntamos la tabla Excel donde se describen estos costes.

Item	Descripción	Unidades	Euros por unidad	Euros totales
HARDWARE				
Hardware	Pentium IV clónico	5	850,00 €	4.250,00 €
	Switch 8 puertos	3	20,00 €	60,00 €
	Cable Ethernet 10Base-T. Par trenzado UTP cat 5. (unidad metro)	10	0,45 €	4,50 €
	Conector RJ-45	20	0,30 €	6,00 €
	Cd´s vírgenes	10	0,60 €	5,38 €
	Total			4.325,88 €
SOFTWARE				
Software	Windows 2003 Server con licencia	1	841,51 €	841,51 €
	ISA Server 2004 Enterprise	1	5.053,32 €	5.053,32 €
	Total			5.894,83 €
MANO DE OBRA				
Ingeniero Técnico	Euros por hora	1100	15,00 €	16500,00 €
COSTE TOTAL				
Coste total proyecto	Proyecto global			**26.720,71 €**

Tabla 21. Coste real del Proyecto

9.3 Líneas de trabajo futuro

La temática de este proyecto es muy extensa y como ya hemos dicho hemos tenido que acotar la cantidad de fuentes de información de las que disponemos y las distintas ramas a las que desencadena un tema como éste. No obstante, los objetivos han sido ambiciosos, pero indudablemente el proyecto deja abiertas varias líneas futuras de investigación. A continuación, se destacan las que a mi juicio tienen una mayor relevancia:

- Estudio de los *Firewalls* con servidores Microsoft.
- Estudio de los *Firewalls* aplicando QoS.
- Estudio de los *Firewalls* utilizando VPNs.

La primera línea de investigación es la que guarda una relación más directa con este proyecto pues la estructura de trabajo sería la misma pero con un escenario en el que los servidores sean Microsoft; de tal manera que se puedan ver las diferencias entre un *Firewall* y otro. Como hemos podido comprobar el análisis de vulnerabilidades de ISA Server ha resultado ser muy parecido a los otros dos, la cual cosa no nos ha permitido comprobar que ISA Server está preparado para proteger la red de determinados EXPLOITS que otros no lo están.

Este proyecto carece de un profundo análisis de los Firewalls cuando se ejecutan funcionalidades de calidad de servicio, pues sería interesante analizar los comportamientos del *Firewall* para poder compararlos a este nivel.

Finalmente, una tercera línea propuesta es la creación de VPNs para poner a prueba el funcionamiento del *Firewall* con este tipo de conexiones.

En definitiva, este proyecto puede servir de base de cualquier otro que tenga como punto de salida algún tema que venga relacionado con cualquiera de estos *Firewalls*, puesto que todo el estudio previo y de comparación está aquí realizado y justificado habiendo partido de cero.

9.4 Conclusiones personales

En cualquier proyecto sea o no para una empresa, uno de los objetivos a los que fundamentalmente se debería llegar siempre es a poder cumplir la planificación temporal que se hace previa al comienzo; siendo que muchas veces aparecen inconvenientes que no se puedan contemplar a priori, resulta difícil poder conseguirlo.

Durante la elaboración de mi proyecto me he encontrado con algunos imprevistos y con las propias dificultades de algunos temas que han supuesto una dedicación más duradera en ello.

El no disponer del material y de las condiciones de trabajo adecuadas en el momento de empezar el proyecto ha provocado que se alargue tres semanas el comienzo del proyecto.

Además, no se preveí correctamente el tiempo que supondría el familiarizarse con las herramientas de *Iptables* y *Packet Filter* y con los respectivos sistemas operativos al tener que hacer múltiples instalaciones en una misma máquina.

Del mismo modo, me ha retrasado el poner por escrito todo lo que se ha hecho en la práctica. He elegido la opción de ir escribiendo en la memoria, ya estructurada, a medida que se hacen las pruebas para no perder detalle, pero esto ha supuesto más cambios en su estructura, puesto que muchas veces según que tipo de información que se va añadiendo supone el variar el enfoque de algunos capítulos.

Por lo que respecta a las pruebas de escaneo de vulnerabilidades las tuve que repetir más de una vez para entender el motivo por el cual eran tan parecidos los resultados de los tres *Firewalls*.

Por otra parte, quiero destacar la parte positiva que me ha aportado este proyecto y que está relacionado con lo mucho que he aprendido; así pues, para la realización de este proyecto y previamente a la realización de las pruebas cuyos resultados se presentan en esta memoria, ha sido necesario un considerable esfuerzo de aprendizaje en los siguientes ámbitos:

En primer lugar se ha estudiado en profundidad el concepto de *Firewall*, su taxonomía y los diferentes tipos de ataques a los que se ven sometidos, pues a lo largo de mis estudios se dan sólo unas pinceladas a lo que *Firewalls* respecta.

Previamente a la configuración de los *Firewalls* fue necesario familiarizarse con el Sistema Operativo con el cual trabajábamos, concretamente con el sistema UNIX OpenBSD. Siendo que no lo había utilizado a lo largo de la carrera lo que emergerse en él conlleva un esfuerzo añadido.

El bloque fundamental de este proyecto lo costituyen la configuración de los *Firewalls* . algunos de los cuales me han llevado más tiempo en su configuración. Para ello ha sido necesario una batería de pruebas continuas por cada una de las modificaciones de reglas, lo que me ha aportado un conocimiento sólido y una agilidad considerable en su configuración.

Respecto a las pruebas de vulnerabilidad realizadas con la herramienta Nessus, ha sido necesario un proceso de aprendizaje acerca del amplio abanico de posibilidades de ataques que esta herramienta ofrece y me ha permitido hacer más hincapié en tomar las medidas de seguridad necesarias, fuera del ámbito de este proyecto, ante los múltiples ataques que pueden llevarse a cabo.

En cuanto a la valoración de cada *Firewall* en posibles escenarios de empresa, indudablemente son muchas las variables que se pueden ofrecer y muchos los condicionantes que pueden existir en estos, lo que dificulta una adecuación perfecta y única de cada uno de ellos con uno de los *Firewalls* aquí estudiados.

No obstante, se espera que toda esta labor sirva de base informativa para ayudar a una empresa a escoger el *Firewall* más adecuado teniendo en cuenta las características de su red.

Llegados a este punto es una honda satisfacción la mía el poder sacar una buena conclusión, cuando me detengo hacer _del mismo modo que con la planificación temporal_ una "revisión de mi conocimiento", pues me doy cuenta de lo que me ha supuesto este proyecto en adquirir nuevas nociones de *Firewalls* y seguridad en redes.

Agradecimientos

Siendo que a veces la vida nos da ciertos contratiempos; siendo que a veces uno tiene días buenos y no tan buenos; siendo que tenemos gente a nuestro alrededor día a día, de alguna manera este trabajo no es sólo mío sino que involucra también a todo aquel que me ha ayudado, por eso he aquí mis agradecimientos.

Para empezar quiero dedicar este proyecto a las personas más importante, mis padres y mis hermanos Daniel y Natalia, porque en todo momento han estado aconsejándome y apoyándome; sin duda no hubiera llegado hasta aquí sin ellos.

Quiero agradecer la ayuda técnica del director de mi proyecto, Marco, y de Joaquim por haberme habilitado un espacio adecuado a las circunstancias del proyecto.

A nivel personal, hago un especial agradecimiento a mi amiga Nadiesda, mi pilar a lo largo de toda la carrera. De igual manera, menciono a Dani y Montse por la agradable compañía prestada a lo largo de todo el cuatrimestre.

Además, quiero destacar el profundo apoyo que he tenido de mis amigas Alum, Anna y Gemma y por supuesto el de él…, Jorge, por ser uno de los protagonistas fundamentales de este año.

Mis agradecimientos van también para Erika, mi soporte más cercano y valioso a lo largo de estos últimos meses de tanto movimiento.

Sin olvidarme de aquellas personas que sin poderlas nombrar a todas, en algún momento me han dado alientos de ánimo. Ya saben quién son...

Gracias a todos de todo corazón. :)

LEYENDA

Red pública de Internet

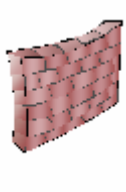
Firewall
Proxy, Bastion

Router

Workstations

ISA Server Array

Servicios



Firewall

GLOSARIO

ALTQ: (*Alternate Queueing*). Plataforma de los sistemas BSD UNIX que proporciona disciplinas de cola y otros componentes QoS.

Array: Grupo de servidores ISA que se usa para implementar tolerancia a fallos y balanceo de carga.

CARP: (***C****ache* ***A****rray* ***R****outing* ***P****rotocol*). En una red con un conjunto de Proxies, CARP busca el camino más efectivo donde se encuentra la respuesta a una petición.

Cookie**:** Cantidad de información que crea un servidor de Internet durante la visita de un usuario a una página Web. Variable que permite al administrador cerrar el servidor en cierto intervalo de tiempo.

Exploits**:** Código escrito con el fin de aprovechar un error de programación para obtener diversos privilegios.

Hashing: Técnica de encriptación que consiste en la conversión de claves textuales en claves numéricas.

ICMP: *Internet Control Message Protocol* Protocolo de control usado en el nivel de red.

Metric: Es la máxima prioridad de la ruta

Microsoft IIS**:** (***I****nternet* ***I****nformation* ***S****erver).* Servidor de páginas Web de Microsoft.

MS SQL Server: Pgrama informático de gestión y administración de bases de datos relacionales basada en el lenguaje SQL (que lanza consultas a una base de datos).

RPC: (***R****emote* ***P****rocedure* ***C****all*). Estándar desarrollado por Sun Microsystems y usado por muchos distribuidores de sistemas UNIX. Se desarrolla en aplicaciones distribuidas y permite que los programas llamen a subrutinas que se ejecutan en un sistema remoto.

RADIUS: (***R****emote* ***A****uthentication* ***D****ial-****I****n* ***U****ser* ***S****ervice*). Sistema de autenticación empleado por la mayoría de proveedores de servicios de Internet (ISPs) si bien no se trata de un estándar oficial. El usuario realiza una conexión a su ISP introduciendo su nombre de usuario y contraseña, información que pasa a un servidor RADIUS que chequeará que la información es correcta y si es así autorizará el acceso al sistema del ISP.

Sniffer:: Persona que recoge datos de la red.

Schedulers: Lo que decide qué colas hay que procesar y en qué orden deben ser procesadas.

BIBLIOGRAFÍA

[1] Brandel D. y Napier R., "Linux 6ª Edición", Domínguez A., Fuentes F. y López E.M., Prentice Hall. Madrid (1999).

[2] Míguez C., Pérez J. y Matas A.M., "La Biblia Hacker", Ediciones Anaya Multimedia. Madrid (2003).

DOCUMENTACIÓN LOCALIZADA EN INTERNET

Capítulo 2. *Firewalls*

- http://www.interhack.net/pubs/fwfaq/
- http://www.pello.info/filez/firewall/iptables.html
- http://www.arcert.gov.ar/cursos/firewalls/ArCERT-Curso_Firewall-2004-2.pdf
- http://es.tldp.org/Manuales-LuCAS/doc-unixsec/unixsec-html/node239.html#dmz
- http://documentos.shellsec.net/otros/Seguridad-basica.pps#19
- http://www.rediris.es/cert/doc/segtcpip/#_Toc12260234

Capítulo 3. Ataques

- http://www.pcm.gob.pe/portal_ongei/seguridad2_archivos/Lib5010/indice.htm
- http://www.idg.es/pcworld/articulo.asp?idart=116336

Capítulo 4. Escenario

- http://acm.asoc.fi.upm.es/documentacion/lpractico/node34.html
- http://www.frikis.org/staticpages/index.php?page=proftpd

Capítulo 5. *Firewall*: *Iptables* sobre Debian Sarge

- http://lists.debian.org/debian-user/2001/12/msg03681.html
- http://www.nautopia.net/archives/es/linux_cortafuegos_e_iptables/iptables/guia_rapida_de_iptables.php
- http://www.tu-chemnitz.de/docs/lindocs/RH73/RH-DOCS/rhl-rg-es-7.3/s1-iptables-options.html
- http://www.linuxdata.com.ar/index.php?idmanual=redfirewall.htm&manuale=1.%20libro%20linux
- http://www.thing.dyndns.org/debian/iptables.htm
- http://ceu.fi.udc.es/~davidfv/documentacion2004/iptables.pdf

- http://mail.linux.org.mx/pipermail/ayuda/2002-August/004408.html
- http://www.espaciolinux.com/artitecid-24.html
- http://iptables-tutorial.frozentux.net/spanish/chunkyhtml/x4996.html
- http://www.seguridad.unam.mx/eventos/admin-unam/FirewallsLinux.pdf

Capítulo 6. *Firewall*: ISA Server Enterprise 2004 sobre Windows 2003 Server

- http://www.microsoft.com/spain/servidores/isaserver/info/features.aspx
- http://www.microsoft.com/spain/servidores/isaserver/info/competitive.aspx#Cost

Capítulo 7. *Firewall*: *Packet Filter* sobre OpenBSD 3.6

- http://www.openbsd.org/faq/es/faq4.html#Start
- http://www.openbsd.org/3.6_packages/i386.html
- http://www.muine.org/~hoang/openpf.html
- http://mirror.etf.bg.ac.yu/openbsd/snapshots/cats/INSTALL.cats
- http://www.muine.org/~hoang/openpf.html
- http://www.bgnett.no/~peter/pf/en/
- http://www.openbsd.org/faq/pf/es/filter.html#example
- http://www.inestable.org/apuntes/openbsd_secure.pdf
- http://www.openbsd.org/faq/pf/queueing.html

Capítulo 8. Balance en base al estudio práctico

- http://www.laflecha.net/canales/seguridad/200410253/
- http://www.bgnett.no/~peter/pf/en/bsdvslinux.html
- http://www.freebsd.org/doc/es_ES.ISO8859-1/articles/explaining-bsd/x99.html

Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANEXOS

TÍTULO DEL TFC: *Firewalls software*: Estudio, instalación, configuración de escenarios y comparativa

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones

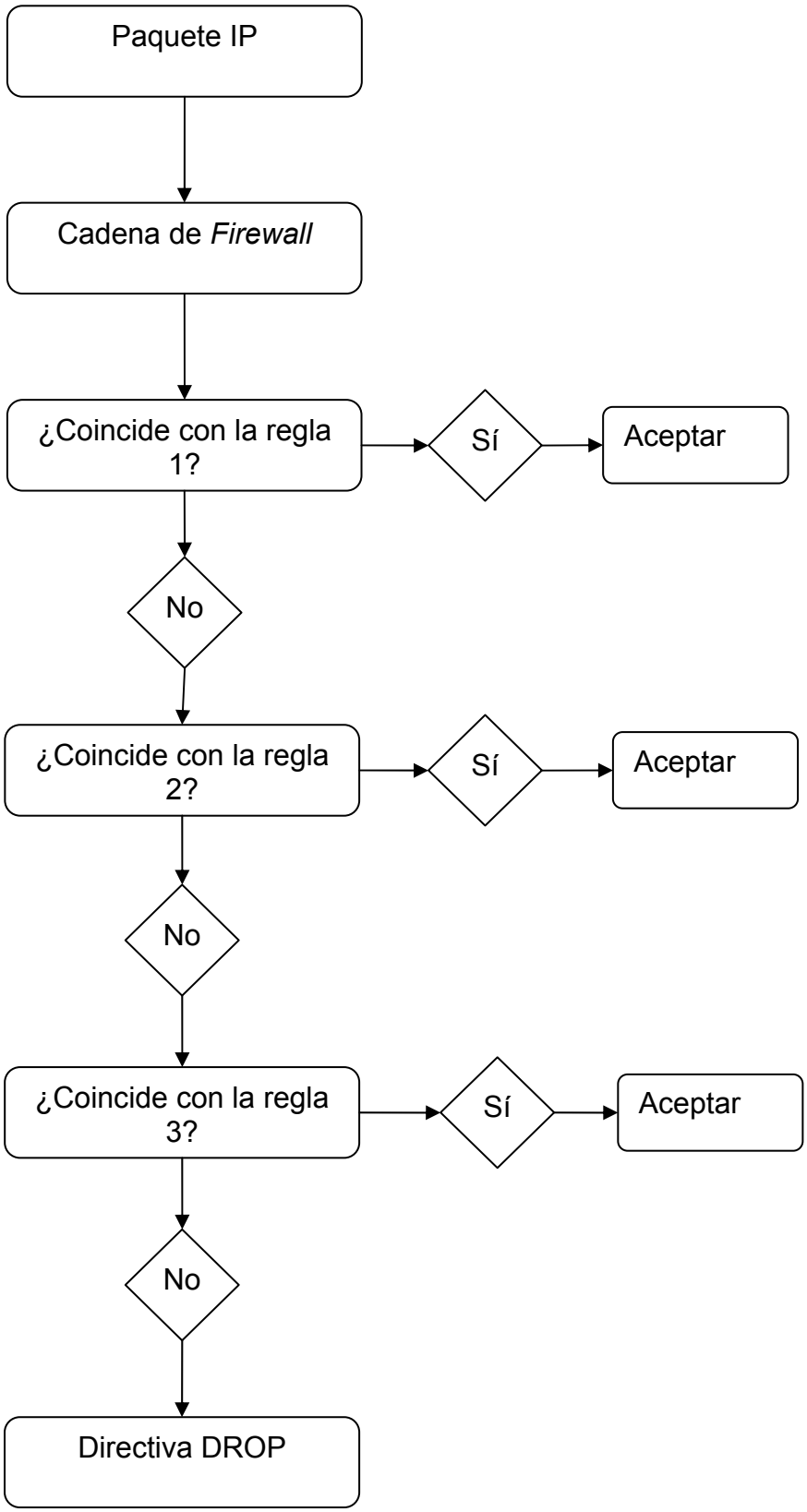
AUTOR: Mònica Ferrer Berbegal

DIRECTOR: Marco Antonio Peña Basurto

DATA: 14 de enero de 2006

ANEXO I: POLÍTICA DE SEGURIDAD

A. Política de negación de servicio



B. Política regida por el *Firewall*

Servidor	Web		Correo		Acceso remoto	Resolución de Nombres	Ficheros	DHCP
Protocolo	**HTTP**	**HTTPS**	**SMTP**	**POP3**	**SSH**	**DNS**	**FTP**	**DHCP**
Puerto #	**80**	**443**	**25**	**110**	**22**	**53**	**20**	**68**
Externa→DMZ	Sí	Sí	Sí	Sí	No	Sí	Sí	No
DMZ→Externa	No	No	No	No	No	Sí	No	No
Interna→DMZ	Sí	Sí	Sí	No	Sí	Sí	Sí	No
DMZ→Interna	No	No	No	No	No	No	No	No
Externa→Interna	No	No	No	No	No	No	No	No
Interna→Externa	Sí	Sí	Sí	Sí	Sí	No	Sí	No
Local Host→DMZ	No	No	No	No	Sí	No	No	No
Local Host→Interna **Interna→Local Host**	No	No	No	No	No	No	No	No
LocalHost→Externa **LocalHost→Externa**	No	No	No	No	No	No	No	No

ANEXO II: FICHEROS DE CONFIGURACIÓN

A. Configuración de las redes

A.1 CONFIGURACIÓN CON DEBIAN SARGE

Firewall

- `/etc/network/interfaces`

```
#The loopback network interface
auto lo
iface lo inet loopback

#The primary network interface (externa)
auto eth0
iface eth0 inet static
        address 192.168.16.214
        netmask 255.255.255.0
        network 192.168.16.0
        gateway 192.168.16.66 #será la salida por defecto

#The internal network interface (LAN privada)
auto eth1
iface eth1 inet static
        address 192.168.100.1
        netmask 255.255.255.0
        network 192.168.100.0
        broadcast 192.168.100.255

#The desmilitarized network interface (DMZ)
auto eth2
iface eth2 inet static
        address 192.168.200.1
        netmask 255.255.255.0
        network 192.168.200.0
        broadcast 192.168.200.255
```

- Tabla de Encaminamiento

Destination	Gateway	Netmask	Iface
192.168.100.0	*	255.255.255.0	eth1
192.168.16.0	*	255.255.255.0	eth0
192.168.200.0	*	255.255.255.0	eth2
default	192.168.16.66	0.0.0.0	eth0

Red interna

- `/etc/network/interfaces`

```
#The loopback network interface
auto lo
iface lo inet loopback

#The primary network interface (interna)
auto eth0
iface eth0 inet static
        address 192.168.100.2
        netmask 255.255.255.0
        network 192.168.100.0
        broadcast 192.168.100.255
```

- Tabla de encaminamiento

La tabla de encaminamiento en este caso es sencillamente que todo lo enviado se remita a la única salida (tarjeta de red) que conecta con el *Firewall* (192.168.100.1), que será la salida por defecto.

Destination	Gateway	Netmask	Iface
192.168.100.0	*	255.255.255.0	eth0
default	192.168.100.1	0.0.0.0	eth0

Red perimétrica (DMZ)

- `/etc/network/interfaces`

```
#The loopback network interface
auto lo
iface lo inet loopback

#The primary network interface (interna)
auto eth0
iface eth0 inet static
        address 192.168.200.2
        netmask 255.255.255.0
        network 192.168.200.0
        broadcast 192.168.200.255
```

Tabla de encaminamiento

De la misma manera que en la máquina del usuario de la LAN privada, al tener una sola tarjeta de red, hay un solo *gateway*, el cual se conecta con el *firewall* (192.168.100.1)

Destination	Gateway	Netmask	Iface
192.168.200.0	*	255.255.255.0	eth0
default	192.168.200.1	0.0.0.0	eth0

A.2 CONFIGURACIÓN DE LAS REDES CON OpenBSD

Firewall

- `/etc/hostname.rl0`

```
inet 192.168.200.1  255.255.255.0  NONE
```

- `/etc/hostname.rl1`

```
inet 192.168.100.1  255.255.255.0  NONE
```

- `/etc/hostname.xl0`

```
inet 192.168.16.214  255.255.255.0  NONE
```

Tabla de encaminamiento

Destination	Gateway	Interface
default	192.168.16.66	xl0
192.168.16/24	link#1	Xl0
192.168.100/24	link#3	rl1
192.168.200/24	link#2	rl0

B. Configuración de los servicios

B.1 CONFIGURACIÓN SERVIDOR DE CORREO. SENDMAIL

- `/etc/mail/sendmail.mc`

#Definimos los macros para identificar la versión de sendmail y la distribución Linux.
#dnl dice al procesador de macros m4 que no procese el rsto de la línea (comentario).
define(`_USE_ETC_MAIL_')dnl
include(`/usr/share/sendmail/cf/m4/cf.m4')dnl
VERSIONID(`$Id: sendmail.mc, v 8.13.4-3 2005-06-03 16:49:22 cowboy Exp $')
OSTYPE(`debian')dnl
DOMAIN(`debian-mta')dnl

FEATURE(`no_default_msa')dnl
Addr=192.168.200.2')dnl
DAEMON_OPTIONS(`Family=inet, Name=MTA-v4, Port=smtp,
Addr=192.168.200.2')dnl
Port=submission, Addr=192.168.200.2')dnl
DAEMON_OPTIONS(`Family=inet, Name=MSP-v4, Port=submission,
Addr=192.168.200.2')dnl Especificamos que escuche no por la interfaz eth1.Por defecto tiene la de localhost o sea que sólo podría recibir mails de la propia máquina.

define(`confPRIVACY_FLAGS',dnl
`needmailhelo,needexpnhelo,needvrfyhelo,restrictqrun,restrictexpand,nobodyreturn,authwarnings')dnl
define(`confCONNECTION_RATE_THROTTLE', `15')dnl
define(`confCONNECTION_RATE_WINDOW_SIZE',`10m')dnl
FEATURE(`access_db', , `skip')dnl
FEATURE(`greet_pause', `1000')dnl 1 seconds
FEATURE(`delay_checks', `friend', `n')dnl
define(`confBAD_RCPT_THROTTLE',`3')dnl
FEATURE(`conncontrol', `nodelay', `terminate')dnl

```
FEATURE(`ratecontrol', `nodelay', `terminate')dnl

LOCAL_CONFIG
EXPOSED_USER(root uucp)dnl # users exempt from masquerading
LOCAL_USER(root)dnl
MASQUERADE_AS(`tfc.es')dnl
FEATURE(`allmasquerade')dnl
FEATURE(`masquerade_envelope')dnl
FEATURE(`nullclient', +)dnl
dnl  #MAILER_DEFINITIONS
dnl #MAILER (‘local’)dnl
dnl #MAILER(‘smtp’)dnl

Cwtfc.es dnl Definimos que tfc.es se trata de un dominio local.
```

- `/etc/mail/local-host-names`

```
DAEMON_OPTIONS ('Port=smtp,Add=192.168.200.2,Name=MTA')
tfc.es
mail.tfc.es
```

- `/etc/mail/relay-domains`

```
tfc.es
mail.tfc.es
gmail.com
```

- `/etc/mail/access`

```
localhost.localdomain      RELAY
localhost                  RELAY
192.168.200.0/24   RELAY  #Subred de la DMZ
192.168.100.0/24   RELAY
gmail.com          RELAY
hotmail            REJECT  #Lo consideramos SPAM
```

B.2 CONFIGURACIÓN SERVIDOR FTP. PROFTPD

- `/etc/proftpd.conf`

```
# /etc/proftpd.conf – This is a Basic ProFTPD configuration file.
#
ServerName          “ftp”
ServerType          inetd   #no es standalone
DeferWelcome        off     #No muestra ningún mensaje
ServerIdent         on      ftp.tfc.es      #Identificador del servidor
```

```
# Esto hará que nuestro servidor ftp sea compatible con casi todos los clientes
ftp.
MultilineRFC2228   on
DefaultServer      on      #Toma las opciones default de un servidor ftp

#el número es el máximo número de segundos que puede estar un cliente en el
servidor sin haber transferencia de información.
TimeoutNoTransfer 600

# el número máximo de segundos que puede estar  cliente-servidor sin recibir
información de una transferencia (atascado).
TimeroutStalled    90

TimeoutIdle        200     #el número máximo de segundos que puede
                           estar un usuario sin hacer nada

DisplayLogin       inicio.msg     #Mensaje de bienvenida
DisplayFirstChdir  .message
ListOptions        "-l"

denyFilter         \*.*/   #Es un filtro de protección para el ProFTPd

Port               21      #Puerto FTP registrado por la IANA[3]

MaxInstances       30      #Número de conexiones al ftp que se pueden hacer
                           a la vez

User               nobody
Group              nogroup

#Mensajes que se mostrarán si la conexión del usuario es correcta o incorrecta
AccessGrantMsg     "Login y Password correctos"
AccessDenyMsg      "Login y password incorrectos"

#Número máximo de clientes que pueden estar a la vez en el servidor
MaxClients         30      "30 clientes como máximo en el servidor"

#Número máximo de clientes  por host
MaxClientsPerHost 12       "12 clientes como máximo en el servidor"
#Número máximo de clientes  por usuario
MaxClientsPerUser  2       "2 conexiones por usuario"

#Le decimos que el directorio del ftp es /home/ftp y a continuación le damos los
permisos de lectura y escritura (4+2) al usuario y a los miembros del grupo.
Estipulamos que el propietario no tiene ningún permiso (0).
        <Directory /home/ftp/>
        Umask              066 066
```

9

[3] IANA: Internet Assigned Numbers Authority

```
        AllowOverwrite      off
        </Directory>
#-----------------------------------------------------------------------------------------
#De la misma manera que para los usuarios privados configuramos el fichero
para clientes anonymous.

        <Anonymous /home/ftp>
AccessGrantMsg          “Bienvenido al servidor FTP”
User                    ftp
Group                   nogroup
UserAlias               anonymous ftp
RequireValidShell       off
MaxClients              30
MaxClientsPerHost       12
MaxClientsPerUser       2

        <Directory /home/ftp/>
        Umask           066 066
        AllowOverwrite off
        </Directory>
</Anonymous>

/etc/resolv.conf

search tfc.es #Dominio de nuestro servidor DNS
nameserver 192.168.200.2 #IP de nuestro servidor
nameserver 194.179.1.100 #IP de nuestro “servidor alternativo”
```

B.3 CONFIGURACIÓN SERVIODR DNS. BIND

- `/etc/bind/named.conf`

```
incluye “/etc/bind/named.conf.options”; //Directiva que dice a named que incluya
otro fichero de zona donde se usa la directiva

// En estas líneas se indica el archivo de la zona raíz donde se encuentran
nombres de los servidores raíces y sus direcciones de IP asociadas
zone "." {
        type hint;
file "/etc/bind/db.root";
};

// Las siguientes líneas indican el archivo de de las zonas: localhost, 127.in-
addr.arpa, 0.in-addr.arpa y 255.in-addr.arpa
zone "localhost"  {
        type master;
file "/etc/bind/db.local";
```

```
};
zone "127.in-addr.arpa"  {
        type master;
        file “/etc/bind/db.127”;
};
zone "0.in-addr.arpa"  {
        type master;
        file “/etc/bind/db.0”;
};
zone "255.in-addr.arpa"  {
        type master;
        file “/etc/bind/db.255”;
};
```

//Archivo donde está configurada la zona de resolución de nombres de nuestro servidor DNS.

```
zone "tfc.es"  {
        type master;
        file “/etc/bind/tfc.es”;
};
```

//Archivo donde está configurada la zona de resolución inversa de nuestro servidor DNS

```
zone "200.192.168.in-addr.arpa"  {
        type master;
        file “/etc/bind/192.168.200.zone”;
};
```

- Ficheros de zonas

Como hemos explicado anteriormente en el apartado de Configuración del servidor DNS (apartado 4.8.2), los ficheros de zonas contienen directivas y registros de recursos.
Las directivas dicen al servidor de nombres que efectúe una determinada acción o que aplique una configuración especial a la zona aunque éstas no son obligatorias. En cambio sí son necesarios los registros de recursos, pues definen los parámetros de la zona asignando una identidad a sistemas particulares en el interior del espacio de nombre de la zona.
A continuación nos disponemos a comentar cada uno de los recursos de los ficheros.

Fichero de resolución DNS

:`/etc/bind/tfc.es`

- TTL: Ajusta el valor *Time to Live (TTL)* predeterminado para la zona. Es el nombre, en segundos, que se da a los servidores de nombres para determinar cuánto tiempo los registros de recursos de la zona serán válidos.

- Serial: Versión del fichero. En nuestro caso es la fecha de configuración (030705). Lo utilizará el servidor esclavo.

Aunque en nuestro caso no tenemos ningún servidor secundario definimos los siguientes tres registros por si en un futuro tuviéramos que configurarlo.

- Refresh: Tiempo (segundos) entre cada actualización de la caché de los servidores secundarios respecto sus servidores primarios.
- Retry: Segundos entre los reintentos de una actualización fallida de un servidor secundario con su primario (604800 s).
- Expire: Tiempo máximo en el que un servidor secundario puede almacenar los datos sin ser actualizados.
- IN: Dirección Internet
- SOA: *Start of Authority* . Describe la zona informada por el fichero. Máquina en la que reside (DMZ.tfc.es) y el responsable (root.tfc.es). Los nombres de las máquinas acaban en . porque no se les debe añadir ningún otro dominio, son nombres completos.
- NS: Servidor de nombres, DMZ.tfc.es es el que tiene autoridad en nuestra zona.

```
IN      NS              DMZ.tfc.es.
```

- A: Registro de dirección que especifica una dirección IP que se debe asignar a un nombre de host. Asociamos el dns con la IP de la máquina 192.168.200.2.

```
DMZ         IN      A           192.168.200.2
```

- MX: Registro Mail Exchange, que identifica dónde deben ser enviados los correos electrónicos del dominio. En nuestro caso es la misma DMZ. A parte se pone el valor de preferencia (10) en el caso que hubiesen más de un servidor de correo; que no es el caso.

```
IN      MX  10          mail.tfc.es
```

- CNAME: Registro del nombre canónico el cual define un alias para un nombre ya conocido. Todas las peticiones enviadas a 'mail', 'www', 'ftp' (nombres de alias) irán dirigidas al host DMZ, en el cual tenemos los tres servidores Web, correo y ftp.

```
www         IN      CNAME       DMZ
mail        IN      CNAME       DMZ
ftp         IN      CNAME       DMZ
```

siendo

```
DMZ         IN      A           192.168.200.2
```

```
;
; Zone file for tfc.es
;
$TTL 3D 259200 seg
@    IN   SOA  DMZ.tfc.es.         root.tfc.es.   (
               030705         ; Serial
```

```
                21600                   ; Refresh  (después de 6
horas)                          3600            ; Retry
(después de una hora)
                604800          ; Expire (después de una
semana)
                86400           )       ; Default TTL (un día
como mínimo)

      IN    NS              DMZ.tfc.es.
            IN    A         192.168.200.2

      IN    MX    10        mail.tfc.es.
            IN    A         192.168.200.2
DMZ         IN    A         192.168.200.2

www         IN    CNAME     DMZ
mail        IN    CNAME     DMZ
ftp         IN    CNAME     DMZ
```

Fichero de resolución inversa

Una resolución de nombres inversa sirve para traducir una dirección IP en un espacio de nombres particular. Para ello se usa el registro PTR con el que se asocia la IP (poniendo el último número de ésta) con el host.

```
      2     IN    PTR   www.tfc.es
```

Siendo el servicio web en 192.168.200.2

```
/etc/bind/192.168.200.zone

;
; BINS reverse data file for 192.168.200.0
; /var/named/192.168.200.zone
;
$TTL 3D // 259200 segundos
@     IN    SOA   DMZ.tfc.es.        root.tfc.es.    (
                  030705              ; Serial
                  21600               ; Refresh  (después de 6 horas)
                  3600                ; Retry   (después de una hora)
                  604800              ; Expire (después de una semana)
                  86400         )     ; Default TTL (un día como mínimo)

      IN    NS    DMZ.tfc.es.

2     IN    PTR   www.tfc.es
2     IN    PTR   mail.tfc.es
2     IN    PTR   ftp.tfc.es
```

B.4 CONFIGURACIÓN SERVIDOR DHCP

- `/etc/dhcpd.conf`

#Tiempo por defecto y máximo en asignar una IP
default-lease-time 600; max lease-time 7200;

option subnet-mask 255.255.255.0;
option broadcast-address 192.168.100.255;

#Gateway→interfaz *Firewall*
option routers 192.168.100.1;

#Servidor DNS que utilizará
option domain-name “tfc.es”;
option domain-name-servers 192.168.200.2, tfc.es.ns.server;

#Definición de la red local con su máscara
subnet 192.168.100.0 netmask 255.255.255.0 {

#Rango de IP’s disponibles que se ofrecerán por DHCP
range 192.168.100.3 192.168.100.50;
}
También podríamos añadir una IP fija a una máquina determinada si fuera necesario.

C. Configuración de los *Firewalls*

C.1 CONFIGURACIÓN FIREWALL IPTABLES

#!/bin/bash

#Cargamos los módulos
/sbin/depmod -a

#Declaración de variables

#Redes
EXTERNA="192.168.16.0/24"
INTERNA="192.168.100.0/24"
DMZ="192.168.200.0/24"

#Interfaces
intEXTERNA="eth0"
intINTERNA="eth1"
intDMZ="eth2"

```
#Servidores DMZ
servidorWEB="192.168.200.2"
servidorFTP="192.168.200.2"
servidorSSH="192.168.200.2"
servidorDNS="192.168.200.2"
servidorMAIL="192.168.200.2"

#Puertos
portHTTP="80"
portHTTPS="443"
portFTP_datos="20"
portFTP_com="21"
portSSH="22"
portSMTP="25"
portPOP3="110"
portDNS="53"

PORTS="1024:65535"

#Borramos todas las reglas anteriores y ponemos a cero los contadores
#Reseteamos las cadenas
iptables -F
#Borramos el contenido de las cadenas definidas por el usuario
iptables -X
#Ponemos a cero los contadores
iptables -Z


#Definimos nuestra política por defecto
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

# NATTING
# Habilitamos la traducción de IP's con el nat
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.16.214

iptables -t nat -A PREROUTING -p tcp -d 192.168.16.214 --dport 80 -j DNAT --
to-destination 192.168.200.2:80
iptables -t nat -A PREROUTING -p tcp -d 192.168.16.214 --dport 443 -j DNAT --
to-destination 192.168.200.2:443
iptables -t nat -A PREROUTING -p tcp -d 192.168.16.214 --dport 20 -j DNAT --
to-destination 192.168.200.2:20
iptables -t nat -A PREROUTING -p tcp -d 192.168.16.214 --dport 21 -j DNAT --
to-destination 192.168.200.2:21
iptables -t nat -A PREROUTING -p tcp -d 192.168.16.214 --dport 22 -j DNAT --
to-destination 192.168.200.2:22
iptables -t nat -A PREROUTING -p tcp -d 192.168.16.214 --dport 53 -j DNAT --
to-destination 192.168.200.2:53
```

```
iptables -t nat -A PREROUTING -p tcp -d 192.168.16.214 --dport 25 -j DNAT --to-destination 192.168.200.2:25
iptables -t nat -A PREROUTING -p tcp -d 192.168.16.214 --dport 110 -j DNAT --to-destination 192.168.200.2:110


# HABILITAMOS SERVICIOS DESDE LA RED INTERNA A LA EXTERNA

# DNS
# Habilitamos únicamente la consulta DNS a la red externa desde la DMZ donde se encuentra nuestro servidor DNS
iptables -A FORWARD -p udp --dport 53 -s $DMZ -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -p udp --sport 53 -d $DMZ -m state --state ESTABLISHED -j ACCEPT


# WEB
# HTTP 80
iptables -A FORWARD -o eth0 -p tcp --dport 80 -s $INTERNA -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --sport 80 -d $INTERNA -m state --state ESTABLISHED -j ACCEPT

# HTTPS 443
iptables -A FORWARD -o eth0 -p tcp --dport 443 -s $INTERNA -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --sport 443 -d $INTERNA -m state --state ESTABLISHED -j ACCEPT


# FTP
# Acceso a los servicios FTP de la red externa, aunque en nuestro escenario no va ser posible.
# Puerto de datos 20
iptables -A FORWARD -o eth0 -p tcp --dport 20 -s $INTERNA -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --sport 20 -d $INTERNA -m state --state ESTABLISHED -j ACCEPT

# Puerto de datos 21
iptables -A FORWARD -o eth0 -p tcp --dport 21 -s $INTERNA -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --sport 21 -d $INTERNA -m state --state ESTABLISHED -j ACCEPT


# SSH
# Habilitamos el acceso remoto de la red interna a la red externa
```

```
iptables -A FORWARD -o eth0 -p tcp --dport 22 -s $INTERNA -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --sport 22 -d $INTERNA -m state --state ESTABLISHED -j ACCEPT


# CORREO
# Habilitamos el servicio SMTP para los usuarios de la red interna
iptables -A FORWARD -o eth0 -p tcp --dport 25 -s $INTERNA -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --sport 25 -d $INTERNA -m state --state ESTABLISHED -j ACCEPT

# Habilitamos el servicio POP3 para los clientes de la red interna
iptables -A FORWARD -o eth0 -p tcp --dport 110 -s $INTERNA -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --sport 110 -d $INTERNA -m state --state ESTABLISHED -j ACCEPT


# HABILITAMOS SERVICIOS DE ENTRADA A LA DMZ DESDE LA EXTERNA (eth0) Y LA INTERNA (eth1)

# DNS
# Acceso al servidor DNS de la dmz
iptables -A FORWARD -p udp --dport 53 -d $servidorDNS -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -p udp --sport 53 -s $servidorDNS -m state --state ESTABLISHED -j ACCEPT

# Acceso desde el localhost del firewall
iptables -A OUTPUT -p udp --dport 53 -d $servidorDNS -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p udp --sport 53 -s $servidorDNS -m state --state ESTABLISHED -j ACCEPT


# WEB
# Acceso al servidor WEB de la DMZ desde la red externa y la interna
iptables -A FORWARD -i eth0 -p tcp --dport 80 -d $servidorWEB -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth1 -p tcp --dport 80 -d $servidorWEB -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth2 -p tcp --sport 80 -s $servidorWEB -m state --state ESTABLISHED -j ACCEPT
```

```
# FTP
# Acceso al servidor FTP de la DMZ desde la red externa y la interna puerto de datos #20
iptables -A FORWARD -i eth0 -p tcp --dport 20 -d $servidorFTP -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth1 -p tcp --dport 20 -d $servidorFTP -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth2 -p tcp --sport 20 -s $servidorFTP -m state --state ESTABLISHED -j ACCEPT

Puerto de comandos #21
iptables -A FORWARD -i eth0 -p tcp --dport 21 -d $servidorFTP -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth1 -p tcp --dport 21 -d $servidorFTP -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth2 -p tcp --sport 21 -s $servidorFTP -m state --state ESTABLISHED -j ACCEPT


# SSH
# Acceso al servidor SSH desde la red interna
iptables -A FORWARD -i eth1 -p tcp --dport 22 -d $servidorSSH -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth2 -p tcp --sport 22 -s $servidorSSH -m state --state ESTABLISHED -j ACCEPT

# Acceso al servidor SSH desde el propio localhost con las cadenas INPUT y OUTPUT
iptables -A OUTPUT -p tcp --dport 22 -d $servidorSSH -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --sport 22 -s $servidorSSH -m state --state ESTABLISHED -j ACCEPT


# CORREO
# Acceso al servidor de correo SMTP de la DMZ desde la red externa y la interna
iptables -A FORWARD -i eth0 -p tcp --dport 25 -d $servidorSMTP -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth1 -p tcp --dport 25 -d $servidorSMTP -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth2 -p tcp --sport 25 -s $servidorSMTP -m state --state ESTABLISHED -j ACCEPT


# PROTECCIONES REDUNDANTES

# Limitamos el número de conexiones de entrada a 5 conexiones por segundo
# Creamos una nueva cadena
```

```
iptables -N syn-flood
iptables -A INPUT -i eth0 -p tcp --syn -j syn-flood
iptables -A syn-flood -m limit --limit 1/s --limit-burst 5 -j RETURN
iptables -A syn-flood -j DROP

# Descartamos los paquetes que no tengan el flag SYN de tcp activado
iptables -A INPUT -i eth0 -p tcp ! --syn -m state --state NEW -j DROP
iptables -A FORWARD -i eth0 -p tcp ! --syn -m state --state NEW -j DROP
```

Después de hacer el primer test de vulnerabilidades añadimos al *Firewall* las siguientes líneas:

```
#Deshabilitamos los SYN.
iptables –A INPUT –p tcp –d 192.168.200.2 –tcp-flags SYN, FIN SYN,FIN –j LOG –m limit –limit 10/m –log-prefix=”boguspacket”
iptables –A INPUT –p tcp –d 192.168.200.2 –tcp-flags SYN, FIN SYN,FIN –j DROP


#Nos aseguramos de cerrar puertos que puedan ser una amenza de seguridad
iptables –A FORWARD –p tcp –dport $portAUTH –m state –state NEW –j DROP
iptables –A FORWARD –p tcp –dport $portNETBIOS_ssn –m state –state NEW –j DROP
iptables –A FORWARD –p tcp –dport $portMICRO_ds –m state –state NEW –j DROP
```

C.2 CONFIGURACIÓN FIREWALL ISA SERVER 2004 ENTERPRISE

Publicación de los servicios

	Servidor Web	Servidor de correo		Servidor SSH	Servidor de FTP	Servidor DNS
General	Servidor WEB dmz	Servidor MAIL SMTP Server	Servidor MAIL POP3 Server	Acceso SSH dmz	Servidor FTP dmz	Acceso DNS dmz
Acción	✅	✅	✅	✅	✅	✅
Desde	Interna Perimetral Externa	Interna Externa	Interna Externa	Interna Local Host	Interna Perimetral Externa	Interna Perimetral Externa
Hacia	192.168.200.2	192.168.200.2	192.168.200.2	192.168.200.2	192.168.200.2	192.168.200.2
Tráfico	http	SMTP Server	POP3 Server	Telnet Server	FTP	DNS Server
Horas	Siempre	Siempre	Siempre	Siempre	Siempre	Siempre
Redes ***Listener***	Externa Interna Perimetral #80	Externa Interna Perimetral	Externa Interna Perimetral	Interna Local Host	Externa Interna Perimetral #21	Interna Perimetral Externa
Nombre público	http://www.tfc.es				http://ftp.tfc.es	
Bridging	http port: 80 ssl port: 443				ftp port: 21	
Usuarios	Todos				Todos	

Política de reglas del *Firewall* y política aplicada por defecto

	Servidor Web		**Servidor de correo**		**Servidor SSH**	**Servidor FTP**		**Servidor DNS**	**Regla por defecto**
General	Acceso web DMZ	Acceso web externo	Acceso mail DMZ	Acceso mail externo	Acceso ssh DMZ	Acceso ftp DMZ	Acceso ftp externo	Acceso dns	Denegación
Acción	✅	✅	✅	✅	✅	✅	✅	✅	⛔
Desde	Interna Perimetral Externa	Interna	Interna Perimetral Externa	Interna	Interna Local Host	Interna Local Host Perimetral Externa	Interna Perimetral	Interna Local Host Perimetral	Todas las redes (incluida Local Host)
Hacia	Perimetral	Externa	Perimetral	Externa	Perimetral Externa	Perimetral	Externa	Externa Perimetral	Todas las redes (incluida Local Host)
Horas	Siempre	Siempre	Siempre	Siempre	Siempre	Siempre	Siempre	Siempre	Siempre
Protocolos	HTTP HTTPS	HTTP HTTPS	POP3 SMTP	POP3 SMTP	SSH	FTP	FTP	DNS	Todo el tráfico
Usuarios	Todos	Todos	Usuarios autenticados	Todos	Usuarios autenticados	Todos	Todos	Todos	Todos
Tipos de contenido	Todos	Todos	Todos	Todos	Todos	Todos	Todos	Todos	Todos

Como podemos ver en la tabla anterior hemos configurado dos reglas de acceso para todos los servicios a excepción del servidor dns y del ssh; una que habilita el tráfico al mismo servicio pero en la red externa y otro a nuestro servidor de la DNS.

Sin embargo, para los servicios de dns y ssh no es necesario crear dos reglas porque en el primero también permitimos el tráfico de la red perimetral a la externa, y para el segundo caso lo deshabilitamos.

C.3 CONFIGURACIÓN *FIREWALL* PACKET FILTER

```
#  Declaración de variables

# Interfaces
int_externa=”xl0”
int_dmz=”rl0”
int_interna=”rl1”

# Redes
red_externa=”192.168.16.0/24”
red_dmz=”192.168.200.0/24”
red_interna=”192.168.100.0/24”

# Definimos lista de servicios de la DMZ a los que se podrá acceder; servicios
TCP y UDP
serv_TCP= { www, ssh, ftp, smtp, pop3 }”
serv_UDP=”{ domain }”
servidorDMZ=”192.168.200.2”


#  1.OPCIONES

# Definimos el comportamiento de la política de negación, para que cuando se
bloquee un paquete TCP se envíe un paquete RSP y para el resto un paquete
ICMP Unrecheable.
set block-policy return


#  2. SCRUB
# Normalización de los paquetes
scrub in all


#  3. NAT
# Servicio NAT en la red DMZ
nat on $int_externa from $int_dmz:network to any → $int_externa

# Servicio NAT en la red interna
nat on $int_externa from $int_interna:network to any → $int_externa


#  4. REGLAS DE FILTRADO
# Establecemos la política por defecto
block log all

# Habilitamos el tráfico en el localhost del firewall
pass quick on lo0 all
```

```
# Habilitamos los servicios TCP y UDP de la DMZ tanto para clientes de la red
interna como de la externa
# Servicios TCP { www, ssh, ftp, smtp, pop3 }
pass in quick on $int_externa inet proto tcp from $red_externa to $servidorDMZ
port $serv_TCP flags S/SA keep state

pass in quick on $int_interna inet proto tcp from $red_interna to $servidorDMZ
port $serv_TCP flags S/SA keep state

#Habilitamos el tráfico de vuelta
pass out quick on $int_dmz inet proto tcp from $red_externa flags S/SA keep
state
pass out quick on $int_dmz inet proto tcp from $red_interna flags S/SA keep
state


# Servicios UDP { domain }
pass in quick on $int_externa inet proto udp from $red_externa to
$servidorDMZ port $serv_UDP keep state
pass in quick on $int_interna inet proto udp from $red_interna to $servidorDMZ
port $serv_UDP keep state



# Habilitamos el tráfico de vuelta para los puertos estándares de los servicios
UDP de la DMZ tanto para la red externa como para la interna
pass out quick on $int_dmz inet proto udp from $red_externa keep state
pass out quick on $int_dmz inet proto udp from $red interna keep state

# Habilitamos el tráfico en los puertos TCP al exterior para los usuarios de la
red interna
pass out quick on $int_externa from any keep state
pass out on &int_externa proto tcp from $red_interna to $red_externa port
$serv_TCP flags S/SA keep state

# Habilitamos además que el servidor DMZ pueda hacer consultas NDS p
pass out on $int_externa proto udp from $servidorDMZ to $red_externa port
$serv_UDP keep state

# Tráfico  de vuelta por el puerto 53 a servidores secundarios

pass in on $int_interna from $int_interna:network to any keep state
pass in on $int_dmz from $int_dmz:network to any keep state

# Antispoof, para las interfaces externa, interna y dmz
antispoof log quick for { $int_externa, $int_interna, $int_dmz }
```

ANEXO III: TABLAS DE DETECCIÓN DE VULNERABILIDADES

A. Tabla de detección de vulnerabilidades *Firewall* Iptables

Servidor Web

Type	**Port**	**Issue and Fix**
Informational	www (80/tcp)	A web server is running on this port Nessus ID : 10330
Informational	www (80/tcp)	An HTTP proxy is running on this port Nessus ID : 10330
Informational	www (80/tcp)	The following directories were discovered: /cgi-bin, /icons, /images While this is not, in and of itself, a bug, you should manually inspect these directories to ensure that they are in compliance with company security standards Other references : OWASP:OWASP-CM-006 Nessus ID : 11032
Informational	www (80/tcp)	The remote webserver supports the TRACE and/or TRACK methods. TRACE and TRACK are HTTP methods which are used to debug web server connections. It has been shown that servers supporting this method are subject to cross-site-scripting attacks, dubbed XST for "Cross-Site-Tracing", when used in conjunction with various weaknesses in browsers. An attacker may use this flaw to trick your legitimate web users to give him their credentials.

		Solution : Add the following lines for each virtual host in your configuration file : RewriteEngine on RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK) RewriteRule .* - [F] See also http://www.kb.cert.org/vuls/id/867593 Risk factor : Medium BID : 9506, 9561, 11604 Nessus ID : 11213
Informational	www (80/tcp)	The remote web server type is : Apache/1.3.33 (Debian GNU/Linux) Solution : You can set the directive 'ServerTokens Prod' to limit the information emanating from the server in its response headers. Nessus ID : 10107

Servidor Correo

Type	Port	Issue and Fix
Informational	smtp (25/tcp)	An SMTP server is running on this port Here is its banner : 220 localhost.localdomain ESMTP Sendmail 8.13.4/8.13.4/Debian-3; Tue, 9 Aug 2005 23:58:17 +0200; (No UCE/UBE) logging access Nessus ID : 10330
Warning	smtp (25/tcp)	The remote SMTP server answers to the EXPN and/or VRFY commands.

		The EXPN command can be used to find the delivery address of mail aliases, or even the full name of the recipients, and the VRFY command may be used to check the validity of an account. Your mailer should not allow remote users to use any of these commands, because it gives them too much information. Solution : if you are using Sendmail, add the option : O PrivacyOptions=goaway in /etc/sendmail.cf. Risk factor : Low CVE : CAN-1999-0531 Nessus ID : 10249
Vulnerability	smtp (25/tcp)	The remote SMTP server did not complain when issued the command : MAIL FROM: |testing This probably means that it is possible to send mail that will be bounced to a program, which is a serious threat, since this allows anyone to execute arbitrary commands on this host. *** This security hole might be a false positive, since *** some MTAs will not complain to this test, but instead *** just drop the message silently Solution : upgrade your MTA or change it. Risk factor : High CVE : CVE-1999-0203 BID : 2308 Nessus ID : 10258

Servidor SSH

Type	Port	Issue and Fix
Informational	ssh (22/tcp)	An ssh server is running on this port Nessus ID : 10330
Informational	ssh (22/tcp)	Remote SSH version : SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4 Remote SSH supported authentication : publickey,keyboard-interactive Nessus ID : 10267
Informational	ssh (22/tcp)	The remote SSH daemon supports the following versions of the SSH protocol : . 1.99 . 2.0 SSHv2 host key fingerprint : 49:5b:16:2f:52:14:c1:85:3d:67:fd:1c:66:1c:cb:5d

Servidor FTP

Type	Port	Issue and Fix
Informational	ftp (21/tcp)	An FTP server is running on this port. Here is its banner : 220 ProFTPD 1.2.10 Server (ftp) Nessus ID : 10330
Warning	ftp (21/tcp)	The remote ProFTPd server is as old or older than 1.2.10

		It is possible to determine which user names are valid on the remote host based on timing analysis attack of the login procedure. An attacker may use this flaw to set up a list of valid usernames for a more efficient brute-force attack against the remote host. Solution : Upgrade to a newer version Risk factor : Low CVE : CAN-2004-1602 BID : 11430 Nessus ID : 15484

Servidor DNS

Type	**Port**	**Issue and Fix**
Informational	domain (53/tcp)	A DNS server is running on this port. If you do not use it, disable it. Risk factor : Low Nessus ID : 11002
Informational	domain (53/tcp)	BIND 'NAMED' is an open-source DNS server from ISC.org. Many proprietary DNS servers are based on BIND source code. The BIND based NAMED servers (or DNS servers) allow remote users to query for version and type information. The query of the CHAOS TXT record 'version.bind', will typically prompt the server to send the information back to the querying source. The remote bind version is : 8.4.6-REL-NOESW Solution : Using the 'version' directive in the 'options' section will block the 'version.bind' query, but it will not log such attempts. Nessus ID : 10028

B. Tabla de detección de vulnerabilidades *Firewall* ISA Server Enterprise 2004

Servidor Web

Type	Port	Issue and Fix
Informational	www (80/tcp)	A web server is running on this port Nessus ID : 10330
Informational	www (80/tcp)	An HTTP proxy is running on this port Nessus ID : 10330
Informational	www (80/tcp)	The following directories were discovered: /cgi-bin, /icons, /images While this is not, in and of itself, a bug, you should manually inspect these directories to ensure that they are in compliance with company security standards Other references : OWASP:OWASP-CM-006 Nessus ID : 11032
Informational	www (80/tcp)	The remote webserver supports the TRACE and/or TRACK methods. TRACE and TRACK are HTTP methods which are used to debug web server connections. It has been shown that servers supporting this method are subject to cross-site-scripting attacks, dubbed XST for "Cross-Site-Tracing", when used in conjunction with various weaknesses in browsers. An attacker may use this flaw to trick your legitimate web users to give him their credentials.

		Solution : Add the following lines for each virtual host in your configuration file : RewriteEngine on RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK) RewriteRule .* - [F] See also http://www.kb.cert.org/vuls/id/867593 Risk factor : Medium BID : 9506, 9561, 11604 Nessus ID : 11213
Informational	www (80/tcp)	The remote web server type is : Apache/1.3.33 (Debian GNU/Linux) Solution : You can set the directive 'ServerTokens Prod' to limit the information emanating from the server in its response headers. Nessus ID : 10107

Servidor Correo

Type	**Port**	**Issue and Fix**
Informational	smtp (25/tcp)	An SMTP server is running on this port Here is its banner : 220 localhost.localdomain ESMTP Sendmail 8.13.4/8.13.4/Debian-3; Tue, 9 Aug 2005 23:58:17 +0200; (No UCE/UBE) logging access Nessus ID : 10330

Informational	smtp (25/tcp)	smtpscan was not able to reliably identify this server. It might be: Sendmail 8.11.0/8.9.3 Sendmail 8.11.6 Sendmail 8.12.3 Sendmail 8.12.11/8.12.11 Sendmail Switch-2.2.0 Sendmail 8.11.6p2/8.11.6 -112- Sendmail 8.12.8/8.11.6 -72- Postfix 2.0.7 Sendmail 8.11.6/8.11.6 -34- Sendmail 8.12.6/8.12.6 Sendmail 8.11.6+Sun/8.9.3 Sendmail 8.11.6p2/8.11.6 Sendmail 8.11.0/8.11.0 Sendmail 8.12.3/8.12.3/SuSE Linux 0.6 Sendmail 8.12.8/8.11.6 -68- Sendmail 8.11.6 Sendmail 8.11.6p2/8.11.6 -145- Sendmail 8.12.2/8.12.2/SideWinder Firewall Sendmail 8.12.8/8.11.4 (2Fast Internet Services) Sendmail 8.11.6p2/8.11.6 -134- The fingerprint differs from these known signatures on 2 point(s) If you known precisely what it is, please send this fingerprint to smtp-signatures@nessus.org : :503:501:501:250:553:250:550:502:252:250:502:502:502:250:250 Nessus ID : 11421

Servidor SSH

Type	**Port**	**Issue and Fix**
Informational	ssh (22/tcp)	An ssh server is running on this port Nessus ID : 10330
Informational	ssh (22/tcp)	Remote SSH version : SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4 Remote SSH supported authentication : publickey,keyboard-interactive

		Nessus ID : 10267
Informational	ssh (22/tcp)	The remote SSH daemon supports the following versions of the SSH protocol : . 1.99 . 2.0 SSHv2 host key fingerprint : 49:5b:16:2f:52:14:c1:85:3d:67:fd:1c:66:1c:cb:5d

Servidor FTP

Type	**Port**	**Issue and Fix**
Informational	ftp (21/tcp)	An FTP server is running on this port. Here is its banner : 220 ProFTPD 1.2.10 Server (ftp) Nessus ID : 10330
Warning	ftp (21/tcp)	The remote ProFTPd server is as old or older than 1.2.10 It is possible to determine which user names are valid on the remote host based on timing analysis attack of the login procedure. An attacker may use this flaw to set up a list of valid usernames for a more efficient brute-force attack against the remote host. Solution : Upgrade to a newer version Risk factor : Low CVE : CAN-2004-1602 BID : 11430 Nessus ID : 15484

Servidor DNS

Type	Port	Issue and Fix
Informational	domain (53/tcp)	A DNS server is running on this port. If you do not use it, disable it. Risk factor : Low Nessus ID : 11002
Informational	domain (53/tcp)	BIND 'NAMED' is an open-source DNS server from ISC.org. Many proprietary DNS servers are based on BIND source code. The BIND based NAMED servers (or DNS servers) allow remote users to query for version and type information. The query of the CHAOS TXT record 'version.bind', will typically prompt the server to send the information back to the querying source. The remote bind version is : 8.4.6-REL-NOESW Solution : Using the 'version' directive in the 'options' section will block the 'version.bind' query, but it will not log such attempts. Nessus ID : 10028

C. Tabla de detección de vulnerabilidades *Firewall* Packet Filter

Servidor Web

Type	**Port**	**Issue and Fix**
Informational	www (80/tcp)	A web server is running on this port Nessus ID : 10330
Warning	www (80/tcp)	It seems that your web server tries to hide its version or name, which is a good thing. However, using a special crafted request, Nessus was able to determine that is is running : Apache/1.3.33 Risk factor : None Solution : Fix your configuration. Nessus ID : 11239

Servidor Correo

Type	**Port**	**Issue and Fix**
Informational	smtp (25/tcp)	An SMTP server is running on this port Here is its banner : 220 localhost.localdomain ESMTP Sendmail 8.13.4/8.13.4/Debian-3; Tue, 9 Aug 2005 23:58:17 +0200; (No UCE/UBE) logging access Nessus ID : 10330
Warning	smtp (25/tcp)	

		The remote SMTP server answers to the EXPN and/or VRFY commands. The EXPN command can be used to find the delivery address of mail aliases, or even the full name of the recipients, and the VRFY command may be used to check the validity of an account. Your mailer should not allow remote users to use any of these commands, because it gives them too much information. Solution : if you are using Sendmail, add the option : O PrivacyOptions=goaway in /etc/sendmail.cf. Risk factor : Low CVE : CAN-1999-0531 Nessus ID : 10249
Vulnerability	smtp (25/tcp)	 The remote SMTP server did not complain when issued the command : MAIL FROM: |testing This probably means that it is possible to send mail that will be bounced to a program, which is a serious threat, since this allows anyone to execute arbitrary commands on this host. *** This security hole might be a false positive, since *** some MTAs will not complain to this test, but instead *** just drop the message silently Solution : upgrade your MTA or change it. Risk factor : High CVE : CVE-1999-0203 BID : 2308 Nessus ID : 10258

Servidor FTP

Type	Port	Issue and Fix
Informational	ftp (21/tcp)	An FTP server is running on this port. Here is its banner : 220 ProFTPD 1.2.10 Server (ftp) Nessus ID : 10330
Warning	ftp (21/tcp)	The remote ProFTPd server is as old or older than 1.2.10 It is possible to determine which user names are valid on the remote host based on timing analysis attack of the login procedure. An attacker may use this flaw to set up a list of valid usernames for a more efficient brute-force attack against the remote host. Solution : Upgrade to a newer version Risk factor : Low CVE : CAN-2004-1602 BID : 11430 Nessus ID : 15484

Servidor SSH

Type	Port	Issue and Fix
Informational	ssh (22/tcp)	An ssh server is running on this port Nessus ID : 10330
Informational	ssh (22/tcp)	Remote SSH version : SSH-2.0-OpenSSH_3.8.1p1 Debian-8.sarge.4

		Remote SSH supported authentication : publickey,keyboard-interactive Nessus ID : 10267
Informational	ssh (22/tcp)	The remote SSH daemon supports the following versions of the SSH protocol : . 1.99 . 2.0 SSHv2 host key fingerprint : 49:5b:16:2f:52:14:c1:85:3d:67:fd:1c:66:1c:cb:5d

Servidor DNS

Type	**Port**	**Issue and Fix**
Informational	domain (53/tcp)	A DNS server is running on this port. If you do not use it, disable it. Risk factor : Low Nessus ID : 11002
Informational	domain (53/tcp)	BIND 'NAMED' is an open-source DNS server from ISC.org. Many proprietary DNS servers are based on BIND source code. The BIND based NAMED servers (or DNS servers) allow remote users to query for version and type information. The query of the CHAOS TXT record 'version.bind', will typically prompt the server to send the information back to the querying source. The remote bind version is : 8.4.6-REL-NOESW Solution : Using the 'version' directive in the 'options' section will block the 'version.bind' query, but it will not log such attempts. Nessus ID : 10028